Focalisation and Classical Realisability (version with appendices)

Guillaume Munch-Maccagnoni

Focalisation and Classical Realisability*

Guillaume Munch-Maccagnoni†

Abstract We develop a polarised variant of Curien and Herbelin's $\bar{\lambda}\mu\tilde{\mu}$ calculus suitable for sequent calculi that admit a focalising cut elimination (i.e. whose proofs are focalised when cut-free), such as Girard's classical logic LC or linear logic. This gives a setting in which Krivine's classical realisability extends naturally (in particular to call-by-value), with a presentation in terms of orthogonality. We give examples of applications to the theory of programming languages.

In this version extended with appendices, we in particular give the two-sided formulation of classical logic with the involutive classical negation. We also show that there is, in classical realisability, a notion of *internal completeness* similar to the one of Ludics.

Contents

*Version with appendices, August 2009. Sections 1–5 appeared in E. Grädel and R. Kahle (Eds.): CSL 2009, LNCS 5771, pp. 409–423, Springer-Verlag. Revised in June 2010 essentially to amend the proof of Example 14.

†Université Paris 7 / INRIA Rocquencourt. Partially funded by INRIA Saclay and the University of Pennsylvania.

1. Introduction

When Curien and Herbelin unveil in [CH00] the computational structure of the sequent calculus, they exhibit a model of computation with a simple interaction between *code* v and *environment* e inside *commands* $c = \langle v \parallel e \rangle$ that recalls abstract machines. This is called the $\bar{\lambda}\mu\tilde{\mu}$ calculus but, following Herbelin [Her08], we will call it system L, as a reference to the tradition of giving sequent calculi names that begin with this letter.

When the proofs from sequent calculus are represented this way, the symmetry of the logic is reflected in the fact that it is the same syntax that describes code (v) and environment (e). In particular, each half of the command can bind the other half with the syntax $\mu x.c'$ (where μ is a binder, and the variable x is bound in the command c' – we in fact merge in a single letter Curien-Herbelin's μ and $\tilde{\mu}$). This leads to computational ambiguities of the following form:

$$c\left[\mu y.c'/x\right] \overset{?}{\leftarrow} \langle \mu x.c \,\|\, \mu y.c' \rangle \overset{?}{\rightarrow} c'\left[\mu x.c/y\right]$$

In the special case of classical logic, x and y can both be fresh in c and c'. The above can therefore lead to the identification of c and c' without any further assumption (*Lafont's critical pair*).

If the goal is to find a computational interpretation of classical sequent calculus, then such ambiguities have to be lifted. Curien and Herbelin [CH00] have achieved an important step in this direction when they have shown that solving the critical pair in favour of the left reduction above yields a computation that corresponds to usual call-by-value (CBV), while the converse choice yields one that corresponds to call-by-name (CBN).

Focalisation Here we tackle this problem from the point of view of focalisation [And92, Gir91]. In the realm of logic programming, Andreoli's focalisation [And92] divides the binary connectives of linear logic (LL) among two groups we shall call the *positives* and the *negatives*. The distinction is motivated by the fact that they can be subject to different assumptions during proof-search. Not long after Andreoli's work, Girard [Gir91] considered focalisation as a way to determinise classical sequent calculus with the classical logic LC, which gives an operational status to these polarities. In the first part of the paper (Section 2) we give a syntax for LC and LL derived from Curien-Herbelin's calculus, the *focalising system* L ($\mathsf{L}_{\mathrm{foc}}$). Despite the age of LC and the proximity of this logic with programming languages, it is the first time that such a

term language is presented, thus answering a question from Girard [Gir91] (see comparison with other works).

The positives are the *tensor* $\otimes$, whose (right-)introduction rule we represent with a pair $(\cdot,\cdot)$, and the *plus* $\oplus$, whose (right-)introduction rules we represent with the two injections $\iota_1(\cdot)$ and $\iota_2(\cdot)$. A formula whose main connective is positive is decomposed hereditarily until an atom or a negative connective is reached. This means that Andreoli's proof-search recipe builds (normal) terms that belong to the following category of *values*:

$$V ::= x \mid t_- \mid (V,V) \mid \iota_1(V) \mid \iota_2(V)$$

where x is a positive variable and the term t_- represents the proof of a negative.

The negatives are the *par* $\parr$ and the *with* &. Their property is that they are invertible, that is they can be decomposed as early as possible during proof-search, a property better reflected with pattern-matching. Keeping such pattern-matching as little bureaucratic as possible, we represent the (right-)introductions of $\parr$ and & respectively with the binders $\mu(x,y).c$ and $\mu\left(\iota_1(x).c \,\middle|\, \iota_2(y).c'\right)$.

The above formulation with values justifies that we see $\otimes$ and $\oplus$ as the connectives for the strict pair and the strict sum (the basic datatypes of ML), as much as the invertibility gives $\parr$ and & a lazy computational behaviour. Focalisation therefore gives classical sequent calculus a crisp computational interpretation that goes past the dichotomy between CBV and CBN that prevails in the works on the duality of computation [CH00, Wad03]: *lazy* and *strict* no longer qualify strategies of evaluation, but connectives of the logic instead, and CBV and CBN become mixed in the same system.

Credit should be given to the authors who first stated the link between focalisation and the values that underpins our syntax. This was not immediate as Girard's formulation of LC uses the *stoup*, a restrained formula in the sequents. Because the relation was "in the air" before being properly written down, it is hard to go back at the roots of the discovery, but we should mention that the work of Curien and Herbelin [CH00] had an early occurrence of values explicitly defined as terms in the stoup, though they were not in the above recursive form. The link was later stated more precisely in the works of Dyckhoff-Lengrand [DL06] and Zeilberger [Zei08].

(As far as classical logic is concerned, we shall in fact present a variant of LC that we call $\mathsf{LK}_{\mathrm{pol}}$ and that, like LL, has the four binary connectives $\otimes, \parr, \&, \oplus$. One finds LC back by using the encodings of $\vee$ and $\wedge$ found in the original article [Gir91].)

Realisability In the second part of the paper (Section 3) we extend Krivine's classical realisability for CBN λ calculus [Kri04] to our setting. In realisability we define for each formula A what it does mean for a term of our language to behave according to (or to *realise*) A – when formulae are seen as specifications for programs. The definition involves orthogonality between terms and is free from any reference to $\mathsf{LK}_{\mathrm{pol}}$ or LL. But the main result (*adequacy*) states that a term of type A also realises A. It therefore provides a justification of the rules of logic.

The commands of L remind the computer scientist of the interaction of a *program* with *data* which is found in the theory of automata. We can therefore make a helpful analogy with finite automata in order to introduce classical realisability.

The analogy replaces the terms of L by words and the states of some NFA $\mathcal{A} = (S, \Sigma, R, s_0, S_F)$. Let us write $\langle \omega \parallel s \rangle$ to symbolise the interaction of a word $\omega \in \Sigma^*$ with $\mathcal{A}$ in some state $s \in S$. One writes $\langle a.\omega \parallel s \rangle \to \left\langle \omega \,\middle\|\, s' \right\rangle$ when $(s,a,s') \in R$ is a transition of the automaton. Orthogonality between words of Σ^* and states of S is defined by taking a set of elements of the form $\langle \omega \parallel s \rangle$ called an *observation* $\perp\!\!\!\perp$. This observation has to be *saturated*, that is to say that if $\langle \omega \parallel s \rangle \to \left\langle \omega' \,\middle\|\, s' \right\rangle \in \perp\!\!\!\perp$, then $\langle \omega \parallel s \rangle \in \perp\!\!\!\perp$. One writes $\omega \perp\!\!\!\perp s$ when $\langle \omega \parallel s \rangle \in \perp\!\!\!\perp$.

For a given observation $\perp\!\!\!\perp$, one then defines $\mathcal{L}^\perp = \{s \in S \mid \forall \omega \in \mathcal{L}, \omega \perp\!\!\!\perp s\}$ for all $\mathcal{L} \subseteq \Sigma^*$ and $\mathcal{S}^\perp = \{\omega \in \Sigma^* \mid \forall s \in \mathcal{S}, \omega \perp\!\!\!\perp s\}$ for all $\mathcal{S} \subseteq S$. Sets of the form $\mathcal{L}^\perp$ or $\mathcal{S}^\perp$ are not ordinary, with the $\mathcal{S}^\perp$ being regular languages. Moreover, if one takes $\perp\!\!\!\perp$ to be the smallest observation for which one has $\varepsilon \perp\!\!\!\perp s_F$ for all $s_F \in S_F$, then $\{s_0\}^\perp$ is the language recognised by $\mathcal{A}$. In addition, the co-linearity of s and s', that is to say $\{s\}^\perp = \{s'\}^\perp$, corresponds to the Nerode equivalence of states s and s'. The equivalence class of s is therefore given by $\{s\}^{\perp\perp}$.

With orthogonality, it is therefore possible to express concisely the main axes of the theory.[1] The intuitions given by orthogonality remain valid with classical realisability, but now we have a much more expressive model of computation that extends λ calculus. Formulae of the logic replace regular expressions, and the sets of terms that realise some formula replace regular languages.

Applications In the third part of the paper (Section 4) come applications. We first show that this method allows us to easily prove properties of normalisation, type safety or parametricity.

$\mathsf{L}_{\mathrm{foc}}$ can be compared to λ calculus when it comes to the study of programming: in particular, the notion of evaluation order is better treated. In support of this argument, we show that classical realisability is discriminating enough to show a clear distinction and relation between the universal quantification coming from proof theory and polymorphism à la ML obtained through *value restriction* (Section 4, "*the two quantifications*"). This issue is indeed related to the order of evaluation imposed by quantifications.

We of course do not claim that second order classical propositional calculus is as such a satisfactory model of computation with respect to the current programming practice. But, as we show, classical realisability accepts in a modular way extensions of the language.

Comparison with Other Works

Danos-Joinet-Schellinx's $\mathsf{LK}^{\eta}_{\mathbf{pol}}$ The paper [DJS95] already considered the four connectives $\otimes, \parr, \&, \oplus$ at the

[1] See Terui's [Ter08] for an earlier appearance of notions of automata theory in an orthogonal setting derived from Ludics.

same time in a derivative of LC called $\mathsf{LK}^{\eta}_{\mathrm{pol}}$ (which was no more a syntax than LC), but we provide an additional justification for this choice: it is the division of the connectives between strict and lazy that justifies the number of connectives. Also, we chose to get rid of the "η-restriction" of $\mathsf{LK}^{\eta}_{\mathrm{pol}}$, hence our choice of the name $\mathsf{LK}_{\mathrm{pol}}$.

The "duality of computation" The works of Curien-Herbelin and Wadler [CH00, Wad03] present a "duality of computation" that appears as the result of a necessary arbitrary choice between CBN and CBV. Laurent established the link with polarities [Lau02], but the duality remained formulated as a dichotomy. On the contrary, $\mathsf{L}_{\mathrm{foc}}$ is a syntax where eager and lazy coexist (as was the case in the non-written two-sided version of LC mentioned by Girard in [Gir91]). The duality of computation is therefore formulated the level of the connectives, as the symmetry between code and environment. This duality is now distinct from the one between positives and negatives.

Comparison with LLP as a candidate syntax for LC The question of giving a representation of LC's proofs, asked by Girard in [Gir91], is ancient. Laurent gave LLP's polarised proof nets as an answer [Lau02]; but it should be said LC's proofs (or equivalently $\mathsf{LK}_{\mathrm{pol}}$'s) are not represented directly in proof nets but through a translation into LLP that introduces modalities. This representation overshadows the notions of evaluation order and values that underlie LC and $\mathsf{LK}_{\mathrm{pol}}$, notions that are important in classical logic as underlined in the syntax we propose.

Ludics We borrow some terminology from Girard's Ludics [Gir01], as well as the the idea of reconstructing types from behaviours. We do not claim however our work should be seen as an alternative version of Ludics, mainly because a syntax based on binders like ours does not offer a proper treatment for the notion of "location" which is prominent in this work.

In addition, we mention the related works of Zeilberger and Terui, from which the present work, which dates back to [MM08], is independent (except for the more recent Section 4, "*The Two Quantifications*", where credit is given). Both Zeilberger [Zei08] and Terui [Ter08] proposed focalised calculi inspired by Girard's Ludics [Gir01]. This prompts a comparison with our proposal.

We share with Zeilberger's *Calculus of Unity* the computational interpretation of the polarities in a calculus that mixes CBV and CBN. Yet Zeilberger's syntax, being of higher order, is not a syntax in the conventional and finitary sense of the word.

Terui's *Computational Ludics* [Ter08] tries to remain closer to Ludics although it does not feature "locations"; and the emphasis is more on the study of complexity. Computational Ludics is fully linear, unlike our setting which can be classical or feature exponentials.

The incentive we have for insisting on using variables and binders, unlike Ludics and like Terui, is that it allows us to remain conventional. For the same reason we chose here to avoid formal pattern-matching and synthetic connectives, unlike Zeilberger and Terui, and we claim to get a syntax that is closer to the tradition of term syntaxes for logic. (Curien and the author's [CMM10] defines however a variant of our syntax that treats patterns as first-class citizens.)

2. Focalising System L

Here we define the syntax and the reduction rules of $\mathsf{L}_{\mathrm{foc}}$.

Syntax *Positive* and *negative variables* are respectively written $x, y, z \ldots$ and $\alpha, \beta, \gamma \ldots$ One defines the sets $\mathcal{T}_+$ and $\mathcal{T}_-$ of the *positive* and *negative terms* t_+ and t_-, as well as the set $\mathcal{C}$ of *commands c:*

$$
\begin{array}{rcll}
\kappa & ::= & \alpha & \mid x \\
t & ::= & t_+ & \mid t_- \\
t_+ & ::= & x & \mid \mu\alpha.c \\
 & & \mid (t,t) & \mid \iota_i(t) (\text{for } i \in \{1,2\}) \qquad (\otimes, \oplus_i) \\
 & & \mid \mu\tilde{}(\kappa).c & \mid \{t\} \qquad (!, \exists) \\
t_- & ::= & \alpha & \mid \mu x.c \\
 & & \mid \mu(\kappa,\kappa).c & \mid \mu\left(\iota_1(\kappa).c \,\middle|\, \iota_2(\kappa).c\right) \qquad (\parr, \&) \\
 & & \mid \tilde{}(t) & \mid \mu\{\kappa\}.c \qquad (?, \forall) \\
c & ::= & \langle t_+ \,\|\, t_- \rangle & \mid \langle t_- \,\|\, t_+ \rangle
\end{array}
$$

with $\mu(\kappa,\kappa').c$ undefined when $\kappa = \kappa'$. Variables that come before a dot in the syntax are bound by the binder μ, and terms and commands are always taken modulo α-equivalence.

$\mathcal{FV}(\cdot)$ denotes the set of the free variables of its argument. $\mathcal{T}_+^0$, $\mathcal{T}_-^0$, $\mathcal{C}^0$ are the sets of the closed terms and commands. In the command $\langle t \,\|\, t' \rangle$, t is called the *counter-term* of t' and t' the counter-term of t.

Formulae Positive *atoms* are written X, Y. Formulae A, B, positive formulae P, Q and negative formulae N, M are given by:

$$
\begin{array}{l}
A ::= P \mid N \\
P ::= X \mid A \otimes A \mid A \oplus A \mid !A \mid \exists X\, A \\
N ::= X^{\perp} \mid A \parr A \mid A \,\&\, A \mid ?A \mid \forall X\, A
\end{array}
$$

(exponentials are given but will only be used for LL). The polarity of a formula is therefore the polarity of its main connective; but it should be noted that it does not introduce constraints of polarity on the logical systems we introduce: our syntax is only polarised at the level of the dynamics, with shifts of polarities left implicit.

One-sided sequents The literature admits two traditions on sequents: Gentzen's two-sided sequents ($\Gamma \vdash \Delta$) and Girard's one-sided sequents ($\vdash \Gamma$). An advantage of the latter is that there is half less rules. The syntax admits both writings and it might be helpful to clarify the link between the two.

Gentzen's tradition makes a distinction between being on the right of the sequent ($\langle t|$) and being on the left of the sequent ($|t\rangle$). In $\langle t \,\|\, u\rangle$, we shall call $\langle t|$ the code and $|u\rangle$ the environment as a legacy of the $\bar{\lambda}\mu\tilde{\mu}$ calculus [CH00] or of Krivine's weak head reduction machine [Kri04]. (For instance, $\langle \iota_1(t)\big|$ shall represent the first injection of a strict sum applied to t, while $\big|\iota_1(t)\rangle$ shall represent the first projection applied to the lazy pair given by the counter-term.)

Girard's tradition, with all the formulae on the right, does not make the distinction between $\langle t|$ and $|t\rangle$. As a consequence the syntax has to be quotiented with a new α-equivalence, $\langle t \,\|\, u\rangle \equiv \langle u \,\|\, t\rangle$. As this paper is in Girard's tradition, this α-equivalence will hold. Reasoning as such *modulo* the left-right symmetry blurs the interpretation in terms of abstract machines, but this simplifies the presentation. But a presentation of the present system with two-sided sequents is available in the Appendix A.

Duality Girard's one-sided tradition requires that we replace the connective of negation $\neg$ by a morphism $\cdot^{\perp}$ on formulae. (Appendix A shows it is of course possible to have this negation in the syntax, and it clearly appears that this negation which changes the polarity is different from the ones that appear in works where there is a choice between CBV and CBN, such as Wadler's Dual Calculus [Wad03].) To each positive formula P (respectively each negative formula N) corresponds a negative *dual* formula $P^{\perp}$ (resp. positive $N^{\perp}$) given by:

$$\begin{array}{cc}
(X)^{\perp} \stackrel{\text{def}}{=} X^{\perp} & (X^{\perp})^{\perp} \stackrel{\text{def}}{=} X \\
(A \otimes B)^{\perp} \stackrel{\text{def}}{=} A^{\perp} \parr B^{\perp} & (A \parr B)^{\perp} \stackrel{\text{def}}{=} A^{\perp} \otimes B^{\perp} \\
(A \oplus B)^{\perp} \stackrel{\text{def}}{=} A^{\perp} \,\&\, B^{\perp} & (A \,\&\, B)^{\perp} \stackrel{\text{def}}{=} A^{\perp} \oplus B^{\perp} \\
(\exists X\, A)^{\perp} \stackrel{\text{def}}{=} \forall X\, A^{\perp} & (\forall X\, A)^{\perp} \stackrel{\text{def}}{=} \exists X\, A^{\perp} \\
(!A)^{\perp} \stackrel{\text{def}}{=} ?A^{\perp} & (?A)^{\perp} \stackrel{\text{def}}{=} !(A^{\perp})
\end{array}$$

One therefore has by definition $A^{\perp\perp} = A$ for each formula A.

Contexts, judgements $\Gamma, \Delta \ldots$ denote *contexts*: sets of elements of the form $x : N$ or $\alpha : P$. The sequents of L_{foc} are judgements of the form:

$$c : (\vdash \Gamma) \qquad \vdash t_+ : P \mid \Gamma \qquad \vdash t_- : N \mid \Gamma$$

In $\vdash t_+ : P \mid \Gamma$ (resp. $\vdash t_- : N \mid \Gamma$), formula P (resp. N) is said to be *principal*. This should not to be confused with the notion of *stoup,* since the latter requires additional constraints of linearity.

Substitution For each formulae A, P and each atom X one defines the formula $A[P/X]$; the important cases are $X[P/X] = P$ and $X^{\perp}[P/X] = P^{\perp}$.

Systems Rules for typing L_{foc} in one-sided MALL, LK_{pol} and LL are given Fig. 1, 2 and 3.

Focalising Weak Head Reduction

We now move on to defining the cut-elimination protocol based on focalisation.

Values *Values* and *positive values* are defined as follows:

$$V ::= V_+ \mid t_- \qquad V_+ ::= x \mid (V, V) \mid \iota_i(V) \mid \mu\tilde{}(\kappa).c \mid \{V\}$$

(It therefore holds, by convention, that any negative term is a value.)

The set of values is written $\mathbb{V}$.

Head Reduction Execution on the calculus is defined as a relation of *head-reduction* on $\mathcal{C}$.

μ-reduction:

$$\langle \mu\alpha.c \,\|\, t_- \rangle \to_{\mu_-} c[t_-/\alpha] \qquad \langle \mu x.c \,\|\, V_+ \rangle \to_{\mu_+} c[V_+/x]$$

β-reduction:

$$\begin{array}{rl}
\langle (V, V') \,\|\, \mu(\kappa, \kappa').c \rangle \to_{\beta} c[V, V'/\kappa, \kappa'] & (\otimes/\parr) \\
\left\langle \iota_i(V) \,\middle\|\, \mu\left(\iota_1(\kappa_1).c_1 \,\middle|\, \iota_2(\kappa_2).c_2\right) \right\rangle \to_{\beta} c_i[V/\kappa_i] & (\oplus_i/\&) \\
\langle \{V\} \,\|\, \mu\{\kappa\}.c \rangle \to_{\beta} c[V/\kappa] & (\exists/\forall) \\
\langle \mu\tilde{}(\kappa).c \,\|\, \tilde{}(V) \rangle \to_{\beta} c[V/\kappa] & (!/?)
\end{array}$$

(In case the polarities of the V's and of the κ's do not match each other, the relation $\to_{\beta}$ is not defined.)

ς-reduction[2]. In case the above rules cannot reduce a command, the following reductions make new cuts appear:

if $t \notin \mathbb{V}$ or $t' \notin \mathbb{V}$ then:

$$\langle (t, t') \,\|\, u_- \rangle \to_{\varsigma} \left\langle t \,\middle\|\, \mu\kappa.\left\langle t' \,\middle\|\, \mu\kappa'.\langle (\kappa, \kappa') \,\|\, u_- \rangle \right\rangle \right\rangle$$

if $t \notin \mathbb{V}$ then:

$$\begin{array}{c}
\langle \iota_i(t) \,\|\, u_- \rangle \to_{\varsigma} \left\langle t \,\middle\|\, \mu x.\langle \iota_i(x) \,\|\, u_- \rangle \right\rangle \\
\langle \{t\} \,\|\, u_- \rangle \to_{\varsigma} \left\langle t \,\middle\|\, \mu x.\langle \{x\} \,\|\, u_- \rangle \right\rangle \\
\langle \tilde{}(t) \,\|\, V_+ \rangle \to_{\varsigma} \left\langle t \,\middle\|\, \mu x.\langle \tilde{}(x) \,\|\, V_+ \rangle \right\rangle
\end{array}$$

Head reduction:

$$\to \quad \stackrel{\text{def}}{=} \quad \to_{\mu_-} \cup \to_{\mu_+} \cup \to_{\beta} \cup \to_{\varsigma}$$

Church-Rosser By definition, $\to$ has no critical pair. This implies the Church-Rosser property when $\to$ is extended to sub-commands. (We have in fact an Orthogonal Pattern Rewrite System, which implies confluence [Nip91].)

Subject reduction Focalising system L enjoys *subject reduction* in both LK_{pol} and LL. (Proof is routine since each connective has a constructor.)

Example We give the example of the implication, writing v the code and e the environment as in Curien-Herbelin's

[2] Terminology borrowed from Wadler [Wad03]. Forbidding non invertible constructors $\otimes, \oplus, \exists, ?$ to contain non-values similarly to [Gir91] would be an alternative to the $\to_{\varsigma}$ reduction, which is therefore available as a convenience. Notice one of course has to arbitrarily decide the evaluation order of the strict pair $(\cdot, \cdot)$.

MALL
Identity

$$\frac{}{\vdash x : P \mid x : P^{\perp}}\ (ax_+) \qquad \frac{}{\vdash \alpha : N \mid \alpha : N^{\perp}}\ (ax_-)$$

$$\frac{c : (\vdash \kappa : A, \Gamma)}{\vdash \mu\kappa.c : A \mid \Gamma}\ (\mu) \qquad \frac{\vdash t : A \mid \Gamma \qquad \vdash u : A^{\perp} \mid \Delta}{\langle t \parallel u \rangle : (\vdash \Gamma, \Delta)}\ (\text{cut})$$

Logic

$$\frac{\vdash t : A \mid \Gamma \qquad \vdash u : B \mid \Delta}{\vdash (t,u) : A \otimes B \mid \Gamma, \Delta}\ (\otimes) \qquad \frac{c : (\vdash \kappa : A, \kappa' : B, \Gamma)}{\vdash \mu(\kappa,\kappa').c : A \parr B \mid \Gamma}\ (\parr)$$

$$\frac{c : (\vdash \kappa : A, \Gamma) \qquad c' : (\vdash \kappa' : B, \Gamma)}{\vdash \mu\left(\iota_1(\kappa).c \,\middle|\, \iota_2(\kappa').c'\right) : A \,\&\, B \mid \Gamma}\ (\&) \qquad \frac{\vdash t : A \mid \Gamma}{\vdash \iota_i(t) : A_1 \oplus A_2 \mid \Gamma}\ (\oplus_i)$$

$$\frac{\vdash t : A[P/X] \mid \Gamma}{\vdash \{t\} : \exists X\, A \mid \Gamma}\ (\exists) \qquad \frac{c : (\vdash \kappa : A, \Gamma)}{\vdash \mu\{\kappa\}.c : \forall X\, A \mid \Gamma}\ (\forall)\ (X \notin \mathcal{FV}(\Gamma))$$

Figure 1: The multiplicative additive linear logic MALL.

$\mathsf{LK}_{\mathrm{pol}}$: MALL + the following structural group:

$$\frac{c : (\vdash \Gamma)}{c : (\vdash \kappa : A, \Gamma)}\ (w) \qquad \frac{c : (\vdash \kappa : A, \kappa' : A, \Gamma)}{c[\kappa/\kappa'] : (\vdash \kappa : A, \Gamma)}\ (c)$$

Figure 2: The constructive classical logic $\mathsf{LK}_{\mathrm{pol}}$.

LL: MALL + the following structural group:

$$\frac{\vdash t : A \mid \Gamma}{\vdash \text{`}(t) : ?A \mid \Gamma}\ (?d) \qquad \frac{c : (\vdash \kappa : A, ?\Gamma)}{\vdash \mu\text{`}(\kappa).c : !A \mid ?\Gamma}\ (!)$$

$$\frac{c : (\vdash \Gamma)}{c : (\vdash x : ?A, \Gamma)}\ (?w) \qquad \frac{c : (\vdash x : ?A, y : ?A, \Gamma)}{c[x/y] : (\vdash x : ?A, \Gamma)}\ (?c)$$

Figure 3: The linear logic LL.

[CH00]. Take:

$$A \to B \stackrel{\text{def}}{=} A^{\perp} \parr B \qquad \lambda\kappa.v \stackrel{\text{def}}{=} \mu(\kappa,\kappa').\langle v \,\|\, \kappa' \rangle \quad (\kappa' \notin \mathcal{FV}(v))$$
$$v \cdot e \stackrel{\text{def}}{=} (v,e) \qquad v\, v' \stackrel{\text{def}}{=} \mu\kappa.\langle v \,\|\, v' \cdot \kappa \rangle \quad (\kappa \notin \mathcal{FV}(v,v'))$$

One has the following derivations:

$$\frac{\vdash v : B \mid \kappa : A^{\perp}, \Gamma}{\vdash \lambda\kappa.v : A \to B \mid \Gamma}\ (\text{abs})$$

$$\frac{\vdash v : A \to B \mid \Gamma \qquad \vdash v' : A \mid \Delta}{\vdash v\, v' : B \mid \Gamma, \Delta}\ (\text{app})$$

We study two particular cases for $A \to B$:
Case A, B negative. This corresponds to CBN. One has, for a positive value E:

$$\langle v\, v' \,\|\, E \rangle \to \langle v \,\|\, v' \cdot E \rangle$$
$$\langle \lambda\alpha.v \,\|\, v' \cdot E \rangle \to \langle v\,[v'/\alpha] \,\|\, E \rangle$$

These are the rules of reduction of a Krivine machine in weak head reduction (as in Krivine's [Kri04]), whose *stacks* are values; or again the rules of the $\bar{\lambda}\mu\tilde{\mu}$ calculus in CBN [CH00].
Case A, B positive. One would expect this to correspond to CBV, and indeed one has:

$$\langle v\, v' \,\|\, e \rangle \to \langle v \,\|\, v' \cdot e \rangle$$

$$\langle \lambda x.v \,\|\, v' \cdot e \rangle \to'^{3} \langle v' \,\|\, \mu x.\langle v \,\|\, e \rangle \rangle$$

(where $\to'$ is the contextual closure of $\to$). Together with $\langle V \,\|\, \mu x.c \rangle \to c\,[V/x]$, this looks like the rules of the CBV $\bar{\lambda}\mu\tilde{\mu}$ calculus. However, this does not correspond to a CBV calculus such as Curien-Herbelin's, since the type of $\to$ is negative and therefore anything of type $P \to Q$ is in $\mathbb{V}$.

Appendix E goes back on the case of implication in a more accurate two-sided setting.

3. Realisability

This section defines a tool for the study of untyped $\mathsf{L}_{\mathrm{foc}}$ based on Krivine's classical realisability for CBN λ calculus [Kri04]. We first define a notion of orthogonality between closed terms.

Definition 1. A subset $\perp\!\!\!\perp$ of $\mathcal{C}^0$ is *saturated* whenever:

$$c \to c', c' \in \perp\!\!\!\perp \Longrightarrow c \in \perp\!\!\!\perp$$

In the rest of the paper, $\perp\!\!\!\perp$ is some saturated subset of $\mathcal{C}^0$ and we say that t is orthogonal to u, and we write $t \perp\!\!\!\perp u$, when $\langle t \,\|\, u \rangle \in \perp\!\!\!\perp$. Because we follow the tradition of single-sided sequents ($\langle t \,\|\, u \rangle \equiv \langle u \,\|\, t \rangle$), one has $t \perp\!\!\!\perp u$ equivalent to $u \perp\!\!\!\perp t$.

Definition 2. Let $T \in \mathcal{P}(\mathcal{T}_+^0)$. One defines:

$$T^\perp = \left\{ t_- \in \mathcal{T}_-^0 \,\middle|\, \forall t_+ \in T, t_+ \perp\!\!\!\perp t_- \right\}$$

Similarly for $T \in \mathcal{P}(\mathcal{T}_-^0)$, one defines:

$$T^\perp = \left\{ t_+ \in \mathcal{T}_+^0 \,\middle|\, \forall t_- \in T, t_+ \perp\!\!\!\perp t_- \right\}$$

A *behaviour*, notation $\mathbf{H}, \mathbf{G} \ldots$, is some subset of $\mathcal{T}_+^0$ or of $\mathcal{T}_-^0$ of the form $T^\perp$. (Terminology is borrowed from Girard's Ludics [Gir01].)

Depending on the polarity of $\emptyset$, one either has $\emptyset^\perp = \mathcal{T}_+^0$ or $\emptyset^\perp = \mathcal{T}_-^0$; disambiguation will be provided by the context.

Proposition 3 (Basic properties of the orthogonal)**.** *Let T and U be two subsets of $\mathcal{T}_+^0$ or $\mathcal{T}_-^0$. (1) One has $T \subset T^{\perp\perp}$. (2) If $T \subseteq U$ then $U^\perp \subseteq T^\perp$. (3) One has $T^{\perp\perp\perp} = T^\perp$. (4) T is a behaviour if and only if $T = T^{\perp\perp}$. (5) If $\mathcal{U}$ is a set of subsets of $\mathcal{T}_+^0$ (resp. of $\mathcal{T}_-^0$), then one has $\left(\bigcup \mathcal{U}\right)^\perp = \bigcap \left\{ T^\perp \,\middle|\, T \in \mathcal{U} \right\}$.*

Behaviours We define for each formula a corresponding behaviour.

Definition 4. *Parameters* $R, S \ldots$ are the members of the set $\Pi \stackrel{\text{def}}{=} \mathcal{P}(\mathcal{T}_+^0 \cap \mathbb{V})$. The language of formulae is extended with parameters: when R is a parameter, R is an atomic positive formula and $R^\perp$ is an atomic negative formula. The systems LK_{pol} and LL are extended with the rule $\dfrac{}{\vdash V_+ : R \mid}$ for each parameter R and each $V_+ \in R$.

Definition 5. Let T and U be two subsets of $\mathcal{T}_+^0$ or $\mathcal{T}_-^0$. One defines:

$$\begin{aligned}
T \times U &= \{(t,u) \mid t \in T, u \in U\} \\
T + U &= \left\{ \iota_1(t) \,\middle|\, t \in T \right\} \cup \left\{ \iota_2(u) \,\middle|\, u \in U \right\} \\
\downarrow T &= \left\{ \{u\} \,\middle|\, u \in T \right\} \\
!T &= \left\{ \mu`(\kappa).c \,\middle|\, V \in T^\perp{}_{\mathbb{V}} \Rightarrow c\,[V/\kappa] \in \perp\!\!\!\perp \right\}
\end{aligned}$$

Definition 6. To each closed positive formula P one associates a behaviour $|P| \in \mathcal{P}(\mathcal{T}_+^0)$ and to each closed negative formula N one associates a behaviour $|N| \in \mathcal{P}(\mathcal{T}_-^0)$. For any term t, one says t *realises* A, and one writes $t \Vdash A$, whenever $t \in |A|$. The definition is given by induction on the size of the formula:

$$\begin{aligned}
|R| &= R^{\perp\perp} & |R^\perp| &= R^\perp \\
|A \otimes B| &= (|A| \times |B|)^{\perp\perp} & |A \parr B| &= (|A^\perp| \times |B^\perp|)^\perp \\
|A \oplus B| &= (|A| + |B|)^{\perp\perp} & |A \,\&\, B| &= (|A^\perp| + |B^\perp|)^\perp \\
|!A| &= (!|A|)^{\perp\perp} & |?A| &= (!|A^\perp|)^\perp \\
|\exists X\, A| &= \left(\bigcup_{R \in \Pi} \downarrow |A\,[R/X]| \right)^{\perp\perp} & |\forall X\, A| &= \left(\bigcup_{R \in \Pi} \downarrow \left|A^\perp\,[R/X]\right| \right)^\perp
\end{aligned}$$

We therefore have by definition that for any closed formula A, one has $|A|^\perp = |A^\perp|$. As a consequence we get an equivalent formulation of realisability, closer to the historical definitions [Kri93]: t realises A if and only if $\forall u\,(u \Vdash A^\perp \Rightarrow t \perp\!\!\!\perp u)$.

Generation lemma What follows is the main lemma required by the main result of the section.

Definition 7.

1. Let $\mathbf{H}$ be a behaviour and $T \subseteq \mathbf{H}$. T generates $\mathbf{H}$ if $\mathbf{H} = T^{\perp\perp}$.
2. Let $\mathbf{H}$ be a behaviour. The set $\mathbf{H}_{\mathbb{V}}$ of the *values of* $\mathbf{H}$ is $\mathbf{H} \cap \mathbb{V}$.

Lemma 8 (Generation)**.** *If A is a closed formula, then $|A|$ is generated by the set of its values.*

The proof requires the lemmas that follow.

Lemma 9. *If $\mathbf{H}$ is a behaviour, then ${\mathbf{H}_{\mathbb{V}}}^{\perp\perp} \cap \mathbb{V} = \mathbf{H}_{\mathbb{V}}$.*

Lemma 10. *Let $\mathbf{H}$ and $\mathbf{G}$ be two behaviours. The following properties hold:*

1. ${\mathbf{H}_{\mathbb{V}}}^{\perp\perp} \times {\mathbf{G}_{\mathbb{V}}}^{\perp\perp} \subseteq (\mathbf{H} \times \mathbf{G})_{\mathbb{V}}{}^{\perp\perp}$;
2. ${\mathbf{H}_{\mathbb{V}}}^{\perp\perp} + {\mathbf{G}_{\mathbb{V}}}^{\perp\perp} \subseteq (\mathbf{H} + \mathbf{G})_{\mathbb{V}}{}^{\perp\perp}$;
3. $\downarrow({\mathbf{H}_{\mathbb{V}}}^{\perp\perp}) \subseteq (\downarrow \mathbf{H}_{\mathbb{V}})^{\perp\perp}$.

Proof. (1) Let $t \in {\mathbf{H}_{\mathbb{V}}}^{\perp\perp}$ and $u \in {\mathbf{G}_{\mathbb{V}}}^{\perp\perp}$; let $v \in (\mathbf{H} \times \mathbf{G})_{\mathbb{V}}{}^\perp$. If $t, u \in \mathbb{V}$, then $(t,u) \in \mathbf{H}_{\mathbb{V}} \times \mathbf{G}_{\mathbb{V}}$ by lemma 9. Yet, by definition, $(\mathbf{H} \times \mathbf{G})_{\mathbb{V}} = \mathbf{H}_{\mathbb{V}} \times \mathbf{G}_{\mathbb{V}}$, hence $(t,u) \perp\!\!\!\perp v$. Otherwise, the result follows by saturation of $\perp\!\!\!\perp$, since $\langle (t,u) \parallel v \rangle$ reduces by $\to_\varsigma$ to an element of $\perp\!\!\!\perp$. (2) and (3): same reasoning. □

Generation lemma. We sketch some key cases of the proof. By induction on the size of A. Case A negative: the result is trivial. Case $A = R$: $|A| = R^{\perp\perp}$ and R is a set of values. Case $A = B \otimes C$: $|B| \times |C|$ is equal to ${|B|_{\mathbb{V}}}^{\perp\perp} \times {|C|_{\mathbb{V}}}^{\perp\perp}$ by induction hypothesis, and is therefore included in $(|B| \times |C|)_{\mathbb{V}}{}^{\perp\perp}$ by lemma 10. Hence $|A|$ is generated by $(|B| \times |C|)_{\mathbb{V}}$. □

Corollary 11 (Substitution)**.** *Let A a formula with $\mathcal{FV}(A)$ of the form $\{X\}$ and P a closed positive formula. $|P|_{\mathbb{V}}$ is a parameter and one has:*

$$|A\,[P/X]| = \left| A\left[|P|_{\mathbb{V}} / X \right] \right|$$

Adequacy lemma The main result of this section affirms that well-typed terms belong to the behaviours described by their types.

Theorem 12 (Adequacy lemma, LK_{pol})**.** *Let c be a command (respectively t a term) typable in LK_{pol}, of type $c : (\vdash \kappa_1 : A_1, \ldots, \kappa_n : A_n)$ (resp. $\vdash t : B \mid \kappa_1 : A_1, \ldots, \kappa_n : A_n$) where the formulae $A_1, \ldots, A_n$ (resp. and B) are closed. For all closed terms $u_1, \ldots, u_n$, if $u_1 \Vdash A_1^\perp, \ldots, u_n \Vdash A_n^\perp$, then $c\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i \right] \in \perp\!\!\!\perp$ (resp. $t\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i \right] \Vdash B$).*

Proof. By induction on the derivation of c and t. The actual induction hypothesis has to be generalised to non-closed formulae, but we can nevertheless sketch the proof with the significant case of activation. Suppose $\vdash \mu\kappa.c : B \mid \Gamma$ comes from $c : (\vdash \kappa : B \mid \Gamma)$. Let $V \in |B^\perp|_{\mathbb{V}}$. One has $\left\langle V \,\middle\|\, \mu\kappa.c\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i \right] \right\rangle \to c\left[V, \overrightarrow{u_i} \middle/ \kappa, \overrightarrow{\kappa_i} \right]$. Yet $c\left[V, \overrightarrow{u_i} \middle/ \kappa, \overrightarrow{\kappa_i} \right] \in \perp\!\!\!\perp$ by induction hypothesis; hence $V \perp\!\!\!\perp \mu\kappa.c\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i \right]$ by saturation. Therefore $\mu\kappa.c\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i \right] \in {|B^\perp|_{\mathbb{V}}}^\perp$, which is equal to $|B|$ by the generation lemma. □

The adequacy lemma holds if one substitutes "LL" for "$\mathsf{LK}_{\mathrm{pol}}$". However, classical realisability would give no quantitative result in relation to linearity.

4. Applications

We show some of the consequences of the adequacy lemma. Proofs are given to show their brevity. In the following, $\vdash$ refers equally to typability in $\mathsf{LK}_{\mathrm{pol}}$ and typability in LL.

Because realisability works with closed terms, we introduce a negative constant, tp (for "top-level"), seen as a pattern matching with no branch, that shall serve as an initial environment which is closed.

Normalisation and type safety The following is an instance of the disjunction property. Such a result usually follows from a cut-elimination theorem and a property of subject reduction.

Example 13. *Let a formula of the form $A_1 \oplus A_2$ and $t \in \mathcal{T}_+^0$ such that $\vdash t : A_1 \oplus A_2$. Then there exists $i \in \{1,2\}$ and a closed value V of the same polarity as A_i such that $\langle t \parallel \mathsf{tp}\rangle \to^* \langle \iota_i(V) \,\|\, \mathsf{tp}\rangle$.*

Proof. Take C the set of the $\langle \iota_i(V) \,\|\, \mathsf{tp}\rangle$ with $i \in \{1,2\}$ and V a closed value of the same polarity as A_i. Take $\perp\!\!\!\perp = \left\{c \in \mathcal{C}^0 \,\middle|\, \exists c_0 \in C, c \to^* c_0\right\}$. For all $V \in |A_i|_{\mathbb{V}}$ one has $\iota_i(V) \perp\!\!\!\perp \mathsf{tp}$, hence $\mathsf{tp} \in \left(|A_1|_{\mathbb{V}} + |A_2|_{\mathbb{V}}\right)^{\perp}$. By proposition 10 and the generation lemma, one therefore has $\mathsf{tp} \Vdash {A_1}^{\perp} \,\&\, {A_2}^{\perp}$. Since the adequacy lemma gives $t \Vdash A_1 \oplus A_2$, one has $t \perp\!\!\!\perp \mathsf{tp}$. □

This example generalises in two directions: (1) With the positive formula left unspecified ($\vdash t : P$), one gets a result of normalisation in head reduction ($\langle t \parallel \mathsf{tp}\rangle \to^* \langle V \parallel \mathsf{tp}\rangle$).[3] (2) The result generalises to other positive formulae: a tensor yields a pair of values, and more generally one has a property of type safety for combinations of $\otimes$ and $\oplus$. This implies type safety for higher-level constructors: a function from A to P supplied with the proper argument yields a result of the expected form. We therefore have an alternative to the traditional acceptation of type safety, where one usually proves subject reduction and other syntactical properties.

Parametricity We prove the uniformity of the universal quantification in an example – which of course generalises.

Example 14. *Let t be a term typable of type $\vdash t : \forall X(X \otimes X \to X \otimes X)$. Let x_1 and x_2 be two positive variables. One has $\left\langle t \,\middle\|\, \{(x_1,x_2)\cdot \mathsf{tp}\}\right\rangle \to^* \left\langle (x_i,x_j) \,\middle\|\, \mathsf{tp}\right\rangle$ for some $i,j \in \{1,2\}$.*

Proof. Indeed if we see x_1 and x_2 as constant positive values[4], then $\perp\!\!\!\perp = \left\{c \in \mathcal{C}^0 \,\middle|\, \exists i,j \in \{1,2\}, c \to^* \left\langle (x_i,x_j) \,\middle\|\, \mathsf{tp}\right\rangle\right\}$ is a (non-empty) observation and $R = \{x_1, x_2\}$ is in Π.

We show that R is complete, which means ${R^{\perp\perp}}_{\mathbb{V}} = R$. We have that $\mu y.\langle (x_1,y) \,\|\, \mathsf{tp}\rangle \in R^{\perp}$. Thus any $V \in {R^{\perp\perp}}_{\mathbb{V}}$ satisfies $\left\langle V \,\middle\|\, \mu y.\langle (x_1,y) \,\|\, \mathsf{tp}\rangle\right\rangle \to^* \left\langle (x_i,x_j) \,\middle\|\, \mathsf{tp}\right\rangle$ for some $i,j \in \{1,2\}$. This implies $V \in \{x_1,x_2\}$, that is to say R is complete.

Thus by Lemma 10 we have that $|R \otimes R| = (R \times R)^{\perp\perp}$, and therefore $\mathsf{tp} \in |R^{\perp} \parr R^{\perp}|$ by definition of $\perp\!\!\!\perp$. We conclude using the adequacy lemma: since we can derive $\left\langle t \,\middle\|\, \{(x_1,x_2)\cdot\alpha\}\right\rangle : (\vdash \alpha : R \otimes R)$, we have that $\left\langle t \,\middle\|\, \{(x_1,x_2)\cdot \mathsf{tp}\}\right\rangle \in \perp\!\!\!\perp$. □

The Two Quantifications

Zeilberger motivated the use of focalisation in order to explain the "imperfections" of realistic typed programming languages such as the value restriction for intersection types in CBV [Zei09]. We show here how classical realisability concisely accounts for such imperfections.

We have shown above that the adequacy lemma by itself gives some form of type safety and normalisation. We can therefore use it as a criterion to test new rules. One of its major advantages is its modularity. Suppose a feature is added to the system under the form of a new connective, with dual inference rules ♡ and ♠. Ensuring that adequacy holds refines into two stages:

(1) Find dual behaviours that correspond to ♡ and ♠, i.e. for which the induction step of adequacy can be shown.

(2) Show that the behaviours of ♡ and ♠ are generated by their values, so that the generation lemma holds.

Modularity comes from the fact that, as one can see, only the rules ♡ and ♠ are involved.

As an example, we apply our method to the possible definitions of ∀ and ∃. The remarks that follow are however general and apply as well to other "intersection types" such as the binary intersection type and first-order universal quantification.

The first definition that comes to mind for the behaviours of the second-order quantifications $\forall X\, A$ and $\exists X\, A$ is the following:

$$|\forall X\, A| \stackrel{?}{=} \bigcap_{R\in\Pi} |A\,[R/X]| \qquad |\exists X\, A| \stackrel{?}{=} \left(\bigcup_{R\in\Pi} |A\,[R/X]|\right)^{\perp\perp}$$

They are dual behaviours by a basic property of the orthogonal, and this definition corresponds to the following inference rules:

$$\frac{\vdash t : A[P/X] \mid \Gamma}{\vdash t : \exists X\, A \mid \Gamma} \qquad \frac{\vdash t : A \mid \Gamma}{\vdash t : \forall X\, A \mid \Gamma}\ (X \notin \mathcal{FV}(\Gamma))$$

Hence this quantification passes the first test. But $|\forall X\, P|$ fails to pass the second test, because of:

Proposition 15. *An intersection of behaviours generated by their values is not generated by its values in general.*

The proof is given in Appendix D. This remark corresponds in particular to the well-known fact that the first implementations of polymorphism in CBV were unsound

[3] As far as strong normalisation is concerned, it should be possible to adapt the technique developed by Lengrand and Miquel [LM08] for a non-polarised and non-confluent symmetric calculus.

[4] Such constants added to the language can be seen as a generalization of the stack constants of Krivine's realizability [Kri04].

in the presence of side-effects (here, control operators of classical logic).[5]

Two distinct solutions that pass the test and that therefore fit the deductive frame of $\mathsf{LK}_{\mathrm{pol}}$ and LL exist.

A first solution: introducing a shift. The impossibility of a positive ∀ is noted by Girard when he develops the denotational semantics of classical logic [Gir91]. The connective ∀ is therefore given the negative polarity in LC. The typing rules of second-order quantification of Fig. 1 introduce to this effect a constructor that forces the polarity. This corresponds to a Curry-style version of the usual quantification ("ΛX") of Church-style system F, which already appeared in Lengrand-Miquel's symmetric and Curry-style adaptation of F_ω [LM08].

A second solution: introducing a value restriction. The second solution restricts the introduction of universal quantification to values, a method found in polymorphism à la ML. It yields quantifications that are different from the first ones, and to make the distinction we shall write them A and E. Value restriction corresponds to the following modification of the above tentative behaviour so that it validates the generation lemma:

$$|\mathsf{A}X\,A| \stackrel{\text{def}}{=} \Big(\bigcap_{R\in\Pi} |A[R/X]|_{\mathbb{V}}\Big)^{\perp\perp}$$

(and dually for E).

As we will see, they are not the usual quantifications, but they are related to ∀ and ∃ as follows: if we consider that $\mu\{\kappa\}.c$ – the constructor for ∀ – corresponds to a shift of polarity at the level of terms that could be made explicit in the types – with an unary connective (written ↑) of the negative polarity – then one has the equality $|\mathsf{A}X{\uparrow}A| = |\forall X\,A|$ (and dually for E).

The adequacy lemma is obtained at the price of the following restriction over the typing rules:

$$\frac{\vdash t : A[P/X] \mid \Gamma}{\vdash t : \mathsf{E}X\,A \mid \Gamma} \qquad \frac{\vdash V : A \mid \Gamma}{\vdash V : \mathsf{A}X\,A \mid \Gamma}\ (X \notin \mathcal{FV}(\Gamma))$$

(Now trying to prove subject reduction for $\mathsf{LK}_{\mathrm{pol}}$ and LL enriched with these rules would be harder, because there are no corresponding constructors in the syntax. With classical realisability, the fact that there are no constructors makes the proof of adequacy even simpler than for ∀ and ∃.)

Comparison of the two solutions Although related, the two kinds of quantification are different, since the latter connective will enjoy paradoxical properties such as the fact that $\mathsf{A}X\,(A \oplus B)$ is the same type as $(\mathsf{A}X\,A) \oplus (\mathsf{A}X\,B)$. This shows that A is not a proper universal quantification for classical logic. (More precisely, we show in Appendix D that for a wide range of observations the equality of behaviours $|\mathsf{A}X\,(A \oplus B)| = |(\mathsf{A}X\,A) \oplus (\mathsf{A}X\,B)|$ holds. By the standards of realisability, this allows one to consider the corresponding type coercions.)

This recalls what Girard called the *"shocking equalities"* of the quantification of Ludics [Gir07]. Now, since our definition of ∀ definitely yields the usual quantification of classical logic, one would tend to question the use of the paradoxical A. Value restriction and its "shocking equalities" are in fact interesting from the computer scientist's point of view, because it gives more sub-typing rules. One example is the intersection type, for which a "shocking" equality such as $\big|(P_1 \cap P_2) \otimes Q\big| = \big|(P_1 \otimes Q) \cap (P_2 \otimes Q)\big|$, that is to say the possibility of introducing coercions between these two types, is desirable.

[5] Specifically, SML/NJ's type system was unsound due to the presence of `call/cc`, as discovered by Harper and Lillibridge in 1991 (http://www.seas.upenn.edu/~sweirich/types/archive/1991/msg00034.html).

5. Conclusion

The present work is not only a concise synthesis but also an extension of distinct works of proof theory: the development of proof syntaxes for sequent calculi initiated by Herbelin and Curien [CH00]; the study of focalisation and polarisation initiated by Andreoli [And92] and Girard [Gir91]; and Krivine's realisability [Kri93, Kri04] that exposes the computational content of proofs.

Yet the result is surprisingly close to the theory of programming languages, as shown by the analogy of Section 1, the status given to values, or the presence of a distinction between "eager" and "lazy" connectives.

Leads for future works are: (1) We would like to study the recent results on the computational content of specific theorems [BD03, Kri04, Kri08] from the point of view of polarisation. (2) We would like to study the practical counterparts to the good theoretical properties of LC's translation of ∧ and ∨ that are exposed in [Gir91]. For instance, it should be possible to base on the present work an extraction procedure for your favourite theorem prover that relies on this translation, which should be compared to extant procedures.

Acknowledgements This work was started and completed at LIX and PPS, but the main part was carried out at Penn. I am grateful to Pierre-Louis Curien and Hugo Herbelin, for helpful interactions and numerous comments around this work, to Stephan Zdancewic and Jeffrey Vaughan for valuable discussions, as well as to the anonymous referees for their comments.

Appendices

In **Section A**, we give the proper two-sided formulation of $\mathsf{LK}_{\mathrm{pol}}$ (and therefore of LC), with the involutive negation of classical logic, as discovered by Girard [Gir91].

In **Section B** we defend a writing convention that augments the conciseness of $\mathsf{L}_{\mathrm{foc}}$.

In **Section C** we explain the computational interpretation of the ⊤ rule as the CBV "toplevel".

In **Section D** we show that the behaviours of classical realisability admit a notion of "internal completeness" similar to the one of Ludics [Gir07].

In **Section E** we show that it is possible to define abstract machines for the CBV and the CBN λ calculus in $\mathsf{L}_{\mathrm{foc}}$ (with local term definitions, i.e. "syntactic sugar").

In **Section F** we compare our formulation of LC with Girard's.

In **Section G** we compare LL with the variant allowed by $\mathsf{L}_{\mathrm{foc}}$.

In **Section H** we detail the proofs for the results given in the main sections.

Acknowledgements These additional sections were written at PPS. I am grateful towards Paul-André Melliès, Stéphane Lengrand and Pierre-Louis Curien for discussions around this work.

A. Two-sided $\mathsf{L}_{\mathbf{foc}}$ and $\mathsf{LK}_{\mathbf{pol}}$ and the classical negation

We give here the two-sided formulation of $\mathsf{L}_{\mathrm{foc}}$ and $\mathsf{LK}_{\mathrm{pol}}$. This makes clearer the fact that there is an involutive negation in classical logic, we shall write $\neg$ in the following.

Having terms on either side of the sequents correspond to making the distinction between $\langle t|$ (code) and $|t\rangle$ (environment). This distinction is necessary for $\mathsf{L}_{\mathrm{foc}}$ to have a meaning in terms of abstract machines, while we see one-sided sequents, which forces the α-equivalence:

$$\langle t \parallel u\rangle \equiv \langle u \parallel t\rangle$$

only as an abstraction on computation where one only cares about the computational flow, without making the distinction between players (or, from the computer scientist's point of view, between inputs and outputs). Thus we see the polarities of logic as a concept distinct from the two players in a game.

A.1. Constructors vs. connectives

In order to avoid confusion due to homonyms, it might be helpful to stress the distinction between *connectives* and *constructors* that exists with two-sided sequents. The constructors $\otimes$, $\parr$, & and $\oplus$ are respectively $(\cdot,\cdot)$, $\mu(\cdot,\cdot).\cdot$, $\mu\left(\iota_1(\cdot).\cdot \,\middle|\, \iota_2(\cdot).\cdot\right)$ and $\iota_i(\cdot)$, while the connectives $\otimes$, $\parr$, & and $\oplus$ correspond respectively to an *eager conjunction*, a *"lazy disjunction"*, a *lazy conjunction* and an *eager disjunction*.

We say that constructors in code position ($\langle\cdot|$) are constructive while those in environment position ($|\cdot\rangle$) are destructive. Thus, regarding for instance the constructor $\otimes$, one has that $\langle(\cdot,\cdot)|$ is constructing the connective $\otimes$ (the eager pair) while $|(\cdot,\cdot)\rangle$ is destructing the connective $\parr$ (the lazy disjunction).

Therefore, the duality $\cdot^{\perp}$ accounts for two notions in $\mathsf{LK}_{\mathrm{pol}}$ which are distinct:

- A duality between focusable and invertible constructors which is defined as a property of good interaction. The constructor $\otimes$ is for instance the dual of the constructor $\parr$, because (t,u) is able to interact with $\mu(\kappa,\kappa').c$.
- A symmetry between connectives known as the "duality of computation". The connective $\oplus$ (the eager disjunction) is for instance the dual of the connective & (the lazy pair), because there is a symmetry between their respective rules of introduction. This corresponds to the fact that the logic is rich enough to give the two players (code, environment) the same set of constructors.

A.2. Negation in $\mathsf{LK}_{\mathbf{pol}}$

In the two-sided setting, negation is represented by a concrete connective instead of the morphism $\cdot^{\perp}$.

The ingredients for this negation have been given in the works of Girard [Gir93] and Danos-Joinet-Schellinx [DJS95], but to our knowledge it has not been given a proper syntactic treatment until now. (For instance, proof nets — because they represent one-sided sequents — are blind about the rules of negation.)

In [Gir93], Girard gives a version of negation in classical logic whose rules send the stoup from one side to the other. In [DJS95], Danos, Joinet and Schellinx identify two symmetric treatments for negation (one can be inverted on the left while the other can be inverted on the right). Then they define negation in $\mathsf{LK}^{\eta}_{\mathrm{pol}}$ as either of these two negations in function of the polarity of the negated formula.

The procedural rules of classical negation we define in the rest of the section corresponds to the above ones of Girard and Danos-Joinet-Schellinx, which happen to be the same.

The positive and the negative negations We introduce two dual connectives that exchange between the left and the right of a sequent: $\neg^{+}$ and $\neg^{-}$.[6] The grammar of the formulae thus becomes:

$$\begin{aligned} A &::= P \mid N \\ X &::= X_{+} \mid X_{-} \\ P &::= X_{+} \mid A \otimes A \mid A \oplus A \mid \exists X\, A \mid \neg^{+} A \\ N &::= X_{-} \mid A \parr A \mid A \,\&\, A \mid \forall X\, A \mid \neg^{-} A \end{aligned}$$

where X_{+} and X_{-} are unrelated atoms. Thus there are now positive and negative substitutions, $A\left[P/X_{+}\right]$ and $A\left[N/X_{-}\right]$. The important cases to define these substitutions are for atoms Y:

$$Y\left[P/X_{+}\right] = \begin{cases} P & \text{if } Y = X_{+} \\ Y & \text{otherwise} \end{cases}$$

$$Y\left[N/X_{-}\right] = \begin{cases} N & \text{if } Y = X_{-} \\ Y & \text{otherwise} \end{cases}$$

Just like one-sided LC defines $\vee$ and $\wedge$ in terms of $\otimes$, $\parr$, & and $\oplus$, we will define $\neg$ in terms of $\neg^{+}$ and $\neg^{-}$. The connectives $\neg^{+}$ and $\neg^{-}$ are respectively positive and negative. In accordance with focalisation, the left-to-right introduction rule of $\neg^{+}$ is focusable, while the left-to-right

[6] They correspond respectively to what Danos, Joinet and Schellinx call the "intuitionistic" and the "classical" negation. We define below the "intuitionistic" and the "classical" negations, which in turn do not correspond to the ones of Danos, Joinet and Schellinx.

introduction rule for $\neg^-$ is invertible. (And conversely for the right-to-left introduction rules.)

The constructors for the left-to-right introduction rules of $\neg^+$ and $\neg^-$ are respectively written $[\cdot]$ and $\mu[\cdot]$ and are understood as follows: for $\left|t_+\right\rangle$ a positive environment, $\left\langle[t_+]\right|$ is positive code; and if $\left|\mu q.c\right\rangle$ is a negative environment (in the informal notation with q a pattern), then $\left\langle\mu[q].c\right|$ is negative code — and symmetrically.

Syntax becomes:

$$\begin{array}{rcll}
\kappa & ::= & \alpha & \mid x \\
t & ::= & t_+ & \mid t_- \\
t_+ & ::= & x & \mid \mu\alpha.c \\
 & & \mid (t,t) & \mid \iota_i(t)(\text{for } i\in\{1,2\}) \qquad (\otimes,\oplus_i) \\
 & & \mid \{t\} & \mid [t] \qquad (\exists,\neg^+) \\
t_- & ::= & \alpha & \mid \mu x.c \\
 & & \mid \mu(\kappa,\kappa).c & \mid \mu\left(\iota_1(\kappa).c \,\middle|\, \iota_2(\kappa).c\right) \qquad (\parr,\&) \\
 & & \mid \mu\{\kappa\}.c & \mid \mu[\kappa].c \qquad (\forall,\neg^-) \\
c & ::= & \langle t_+ \,\|\, t_-\rangle & \mid \langle t_- \,\|\, t_+\rangle
\end{array}$$

Though we make the distinction between $\langle t|$ and $|t\rangle$ it is the same syntax that define both the code and the environment, thus allowing some concision in the definition. It would however be meaningless to allow variables ($\langle\kappa|$) and co-variables ($|\kappa\rangle$) to be mixed, e.g. to allow κ to be in one command a variable and in another one a co-variable. We shall therefore mark co-variables with a *bar* ($|\bar\kappa\rangle$), and outcast commands in which some κ appears both with and without a bar. Curien-Herbelin's "call/cc" binder $\mu\kappa.c$ will therefore be written $\left\langle\mu\bar\kappa.c\right|$ in our syntax, while the "let ... in" binder is written $\left|\mu\kappa.c\right\rangle$.[7]

Values become:

$$\begin{aligned}
V &::= V_+ \mid t_- \\
V_+ &::= x \mid (V,V) \mid \iota_i(V) \mid \{V\} \mid [V]
\end{aligned}$$

The new reduction rules are as follows:

$$\begin{aligned}
\left\langle[V]\,\middle\|\,\mu[\bar\kappa].c\right\rangle &\to_\beta c\,[V/\bar\kappa] \\
\left\langle\mu[\kappa].c\,\middle\|\,[V]\right\rangle &\to_\beta c\,[V/\kappa]
\end{aligned}$$

[7] But if one wants to remain closer to the formulation of $\bar\lambda\mu\tilde\mu$ with distinct classes v and e, then one needs to have two ways of writing each constructor. One therefore distinguishes variables κ, co-variables $\bar\kappa$, code v and environment e in a following lengthy way:

$$\begin{array}{rcll}
\kappa & ::= & \alpha & \mid x \\
\bar\kappa & ::= & \bar\alpha & \mid \bar x \\
v & ::= & v_+ & \mid v_- \\
v_+ & ::= & x & \mid \mu\tilde\alpha.c \\
 & & \mid (v,v) & \mid \iota_i(v)(\text{for } i\in\{1,2\}) \qquad (\otimes,\oplus_i) \\
 & & \mid \{v\} & \mid \lfloor e\rfloor \qquad (\exists,\neg^+) \\
v_- & ::= & \alpha & \mid \mu\bar x.c \\
 & & \mid \mu[\bar\kappa,\bar\kappa].c & \mid \mu\left[\pi_1\cdot\bar\kappa.c \,\middle|\, \pi_2\cdot\bar\kappa.c\right] \qquad (\parr,\&) \\
 & & \mid \mu\{\bar\kappa\}.c & \mid \mu\lceil\bar\kappa\rceil.c \qquad (\forall,\neg^-) \\
e & ::= & e_+ & \mid e_- \\
e_+ & ::= & \bar x & \mid \tilde\mu\alpha.c \\
 & & \mid [e,e] & \mid \pi_i\cdot e\ (\text{for } i\in\{1,2\}) \qquad (\parr,\&_i) \\
 & & \mid \{e\} & \mid \lceil v\rceil \qquad (\forall,\neg^-) \\
e_- & ::= & \tilde\alpha & \mid \tilde\mu x.c \\
 & & \mid \tilde\mu(\kappa,\kappa).c & \mid \tilde\mu\left(\iota_1(\kappa).c \,\middle|\, \iota_2(\kappa).c\right) \qquad (\otimes,\oplus) \\
 & & \mid \tilde\mu\{\bar\kappa\}.c & \mid \tilde\mu\lfloor\kappa\rfloor.c \qquad (\exists,\neg^+) \\
c & ::= & \langle v_+ \,\|\, e_-\rangle & \mid \langle v_- \,\|\, e_+\rangle
\end{array}$$

But does such a waste of paper make anything clearer? One can doubt so.

$$\begin{aligned}
\langle[t]\,\|\,u\rangle &\to_\varsigma \left\langle\mu\bar x.\langle[\bar x]\,\|\,u\rangle\,\middle\|\,t\right\rangle \qquad (t\notin\mathbb{V}) \\
\langle u\,\|\,[t]\rangle &\to_\varsigma \left\langle t\,\middle\|\,\mu x.\langle u\,\|\,[x]\rangle\right\rangle \qquad (t\notin\mathbb{V})
\end{aligned}$$

(The remaining rules of reduction for two-sided L_{foc} can be inferred from the ones of Section 2, each reduction rule of one-sided L_{foc} yielding two symmetric rules in the two-sided version.)

Two-sided LK_{pol} is given fig. 4.

Negation in LC: an involutive negation A connective of negation $\neg$ for two-sided LK_{pol} (as well as for two-sided LC, which is a fragment of the latter) is then defined as follows:

Definition 16. The formula $\neg A$ is defined in function of the polarity of A as follows:

A	P	N
$\neg A$	$\neg^- P$	$\neg^+ N$

This defines a negation that exchanges the polarity of the formulae. As a consequence, at the level of constructors, the rules of negation do *not* change the polarity of terms. This is why we can have an isomorphism between A and $\neg\neg A$.

Let us define the equivalence on terms that will allow us to state this isomorphism. It is the analogue of the $\beta\eta$ equivalence of the λ calculus.

Definition 17 ($t\simeq u$)**.** One considers the following "observational" rules:

$$\begin{aligned}
\mu\kappa.\langle\kappa\,\|\,t\rangle &\to_\eta t \\
\mu\bar\kappa.\langle t\,\|\,\bar\kappa\rangle &\to_\eta t \\
\mu[\kappa].\langle t\,\|\,[\kappa]\rangle &\to_\eta t \\
\mu[\bar\kappa].\langle[\bar\kappa]\,\|\,t\rangle &\to_\eta t
\end{aligned}$$

The relation $\simeq$ on terms is defined as the smallest equivalence relation such that:

- If u is obtained from t by the application of $\to$ on a sub-command of t, then $t\simeq u$,
- If u is obtained from t by the application of $\to_\eta$ on t or one of its sub-terms, then $t\simeq u$.

In the statement of $\neg\neg A\simeq A$ that follows, the categorical composition is of a syntactical nature, like substitution, rather than of a more semantical nature, like a cut. In fact, when composition is interpreted by a cut, one fails to account for logics that have two polarities, like LC or LK_{pol}. This is shown by the following remark:

Remark 18. Suppose we define the operation $\circ$ on commands $c:(\kappa_1:A\vdash\bar\kappa_2:B)$ and $c':(\kappa_3:B\vdash\bar\kappa_4:C)$ with:

$$c'\circ c=\left\langle\mu\bar\kappa_2.c_1\,\middle\|\,\mu\kappa_3.c_2\right\rangle$$

Then take $c_1:(\kappa:A\vdash\bar\alpha:N)$ and $c_2:(x:N\vdash\bar y:P)$ and $c_3:(\beta:P\vdash\bar\kappa':B)$. If $\bar\alpha\notin\mathcal{FV}(c_1)$ and $\beta\notin\mathcal{FV}(c_3)$, then one has:

$$(c_1\circ c_2)\circ c_3\simeq c_3$$

$\mathsf{LK}_{\mathrm{pol}}$ (Two-sided)

Identity

$$\frac{}{\mid \bar{\kappa} : A \vdash \bar{\kappa} : A}\ (ax\vdash) \qquad \frac{}{\kappa : A \vdash \kappa : A \mid}\ (\vdash ax)$$

$$\frac{c : (\Gamma \vdash \bar{\kappa} : A, \Delta)}{\Gamma \vdash \mu\bar{\kappa}.c : A \mid \Delta}\ (\vdash \mu) \qquad \frac{c : (\Gamma, \kappa : A \vdash \Delta)}{\Gamma \mid \mu\kappa.c : A \vdash \Delta}\ (\mu \vdash)$$

$$\frac{\Gamma \vdash t : A \mid \Delta \qquad \Gamma' \mid u : A \vdash \Delta'}{\langle t \parallel u \rangle : (\Gamma, \Gamma' \vdash \Delta, \Delta')}\ \textbf{(cut)}$$

Structure

$$\frac{c : (\Gamma \vdash \Delta)}{c : (\Gamma \vdash \bar{\kappa} : A, \Delta)}\ (\vdash w) \qquad \frac{c : (\Gamma \vdash \Delta)}{c : (\Gamma, \kappa : A \vdash \Delta)}\ (w \vdash)$$

$$\frac{c : (\Gamma \vdash \bar{\kappa} : A, \bar{\kappa}' : A, \Delta)}{c\left[\bar{\kappa}/\bar{\kappa}'\right] : (\Gamma \vdash \bar{\kappa} : A, \Delta)}\ (\vdash c) \qquad \frac{c : (\Gamma, \kappa : A, \kappa' : A \vdash \Delta)}{c\left[\kappa/\kappa'\right] : (\Gamma, \kappa : A \vdash \Delta)}\ (c \vdash)$$

(and similar rules for $\langle t|$ and $|t\rangle$.)

Logic

$$\frac{\Gamma \mid t : A \vdash \Delta}{\Gamma \vdash [t] : \neg^+ A \mid \Delta}\ (\vdash \neg^+) \qquad \frac{c : (\Gamma \vdash \bar{\kappa} : A, \Delta)}{\Gamma \mid \mu[\bar{\kappa}].c : \neg^+ A \vdash \Delta}\ (\neg^+ \vdash)$$

$$\frac{c : (\Gamma, \kappa : A \vdash \Delta)}{\Gamma \vdash \mu[\kappa].c : \neg^- A \mid \Delta}\ (\vdash \neg^-) \qquad \frac{\Gamma \vdash t : A \mid \Delta}{\Gamma \mid [t] : \neg^- A \vdash \Delta}\ (\neg^- \vdash)$$

$$\frac{\Gamma \vdash t : A \mid \Delta \qquad \Gamma' \vdash u : B \mid \Delta'}{\Gamma, \Gamma' \vdash (t, u) : A \otimes B \mid \Delta, \Delta'}\ (\vdash \otimes) \qquad \frac{c : (\Gamma, \kappa : A, \kappa' : B \vdash \Delta)}{\Gamma \mid \mu(\kappa, \kappa').c : A \otimes B \vdash \Delta}\ (\otimes \vdash)$$

$$\frac{c : (\Gamma \vdash \bar{\kappa} : A, \bar{\kappa}' : B, \Delta)}{\Gamma \vdash \mu(\bar{\kappa}, \bar{\kappa}').c : A \parr B \mid \Delta}\ (\parr \vdash) \qquad \frac{\Gamma \mid t : A \vdash \Delta \qquad \Gamma' \mid u : B \vdash \Delta'}{\Gamma, \Gamma' \mid (t, u) : A \parr B \vdash \Delta, \Delta'}\ (\parr \vdash)$$

$$\frac{c : (\Gamma \vdash \bar{\kappa} : A, \Delta) \qquad c' : (\Gamma \vdash \bar{\kappa}' : B, \Delta)}{\Gamma \vdash \mu\left(\iota_1(\bar{\kappa}).c \,\middle|\, \iota_2(\bar{\kappa}').c'\right) : A \,\&\, B \mid \Delta}\ (\vdash \&) \qquad \frac{\Gamma \mid t : A_i \vdash \Delta}{\Gamma \mid \iota_i(t) : A_1 \,\&\, A_2 \vdash \Delta}\ (\&_i \vdash)$$

$$\frac{\Gamma \vdash t : A_i \mid \Delta}{\Gamma \vdash \iota_i(t) : A_1 \oplus A_2 \mid \Delta}\ (\vdash \oplus_i) \qquad \frac{c : (\Gamma, \kappa : A \vdash \Delta) \qquad c' : (\Gamma, \kappa' : B \vdash \Delta)}{\Gamma \mid \mu\left(\iota_1(\bar{\kappa}).c \,\middle|\, \iota_2(\bar{\kappa}').c'\right) : A \oplus B \vdash \Delta}\ (\oplus \vdash)$$

$$\frac{\Gamma \vdash t : A[B/X] \mid \Delta}{\Gamma \vdash \{t\} : \exists X\, A \mid \Delta}\ (\vdash \exists) \qquad \frac{c : (\Gamma, \kappa : A \vdash \Delta)}{\Gamma \mid \mu\{\kappa\}.c : \exists X\, A \vdash \Delta}\ (\exists \vdash)\ (X \notin \mathcal{FV}(\Gamma))$$

$$\frac{c : (\Gamma \vdash \bar{\kappa} : A, \Delta)}{\Gamma \vdash \mu\{\bar{\kappa}\}.c : \forall X\, A \mid \Delta}\ (\vdash \forall)\ (X \notin \mathcal{FV}(\Gamma)) \qquad \frac{\Gamma \mid t : A[B/X] \vdash \Delta}{\Gamma \mid \{t\} : \forall X\, A \vdash \Delta}\ (\forall \vdash)$$

In the rules $(\vdash \exists)$ and $(\forall \vdash)$, X and B have to share the same polarity.

Figure 4: The constructive classical logic $\mathsf{LK}_{\mathrm{pol}}$ in a two-sided setting.

$$c_1 \circ (c_2 \circ c_3) \simeq c_1$$

$\circ$ is therefore not associative modulo $\simeq$, since one has in general $c_1 \not\simeq c_3$.

We also stress the distinction between terms (or commands) and morphisms for the following similar reason:

Remark 19. For a command c and two terms $t = \mu\alpha.c_1$ and $u = \mu\beta.c_2$ with $\alpha, \beta \notin \mathcal{FV}(c_1, c_2)$, one has:

$$\langle t \,\|\, \mu x.\langle u \,\|\, \mu y.c\rangle\rangle \simeq c_1$$
$$\langle u \,\|\, \mu y.\langle t \,\|\, \mu x.c\rangle\rangle \simeq c_2$$

It implies that one does not have in general:

$$\langle t \,\|\, \mu x.\langle u \,\|\, \mu y.c\rangle\rangle \simeq \langle u \,\|\, \mu y.\langle t \,\|\, \mu x.c\rangle\rangle$$

But one has:

$$\langle x \,\|\, \mu x.\langle y \,\|\, \mu y.c\rangle\rangle \simeq \langle y \,\|\, \mu y.\langle x \,\|\, \mu x.c\rangle\rangle$$

This means that $(x, y) \mapsto c$ is not compatible with $\simeq$, i.e. categories that yield denotational models for $\mathsf{LK}_{\mathrm{pol}}$ are not cartesian. This point was already underlined by Girard [Gir91] in the case of LC.

This is why we present morphisms as functions and $\circ$ as the composition of functions. But we close the categorical digression, since the proposition below, even though it takes a categorical style, can be understood independently from categorical considerations.

Proposition 20 ($\neg\neg A \simeq A$)**.** *Define the following morphisms on terms:*[8]

$$f(t) = \begin{cases} [[t]] & \text{if } t \in \mathcal{T}_+ \\ \mu[[\bar{x}]].\langle t \,\|\, \bar{x}\rangle & \text{if } t \in \mathcal{T}_- \end{cases}$$
$$g(u) = \begin{cases} \mu\bar{\alpha}.\langle u \,\|\, \mu[[x]].\langle x \,\|\, \bar{\alpha}\rangle\rangle & \text{if } t \in \mathcal{T}_+ \\ \mu\bar{x}.\langle u \,\|\, [[\bar{x}]]\rangle & \text{if } t \in \mathcal{T}_- \end{cases}$$

One has for all A the following derivations, with κ a variable that has the same polarity as A:

$$\kappa : A \vdash f(\kappa) : \neg\neg A \,|$$
$$\kappa : \neg\neg A \vdash g(\kappa) : A \,|$$

One has for all terms t:

$$g \circ f(t) \simeq t$$
$$f \circ g(t) \simeq t$$

Proof. First observe that the relation $\simeq$ yields ς and η rules extended to compound patterns, i.e. if $t_+ \notin \mathbb{V}$:

$$\begin{aligned} &\mu\kappa.\langle [[t_+]] \,\|\, u\rangle \\ &\quad \simeq \mu\kappa.\langle \mu\bar{x}.\langle [\bar{x}] \,\|\, u\rangle \,\|\, [t_+]\rangle \\ &\quad \simeq \mu\kappa.\langle t_+ \,\|\, \mu y.\langle \mu\bar{x}.\langle [\bar{x}] \,\|\, u\rangle \,\|\, [y]\rangle\rangle \\ &\quad \simeq \mu\kappa.\langle t_+ \,\|\, \mu y.\langle [[y]] \,\|\, u\rangle\rangle \end{aligned}$$

$$\begin{aligned} &\mu[[x]].\langle [[x]] \,\|\, u\rangle \\ &\quad = \mu[\bar{y}].\langle \mu[x].\langle [[x]] \,\|\, u\rangle \,\|\, \bar{y}\rangle \\ &\quad \simeq \mu[\bar{y}].\langle \mu[x].\langle \mu\bar{z}.\langle [\bar{z}] \,\|\, u\rangle \,\|\, [x]\rangle \,\|\, \bar{y}\rangle \\ &\quad \simeq \mu[\bar{y}].\langle \mu\bar{z}.\langle [\bar{z}] \,\|\, u\rangle \,\|\, \bar{y}\rangle \\ &\quad \simeq \mu[\bar{y}].\langle [\bar{y}] \,\|\, u\rangle \\ &\quad \simeq u \end{aligned}$$

Now, for the case $t_+ \notin \mathbb{V}$, one has:

$$\begin{aligned} g \circ f(t_+) &= \mu\bar{\alpha}.\langle [[t_+]] \,\|\, \mu[[x]].\langle x \,\|\, \bar{\alpha}\rangle\rangle \\ &\simeq \mu\bar{\alpha}.\langle t_+ \,\|\, \mu y.\langle [[y]] \,\|\, \mu[[x]].\langle x \,\|\, \bar{\alpha}\rangle\rangle\rangle \\ &\simeq \mu\bar{\alpha}.\langle t_+ \,\|\, \mu y.\langle y \,\|\, \bar{\alpha}\rangle\rangle \\ &\simeq t_+ \end{aligned}$$

$$\begin{aligned} f \circ g(t_+) &= [[g(t_+)]] \\ &\simeq \mu\bar{\beta}.\langle [[g(t_+)]] \,\|\, \bar{\beta}\rangle \\ &\simeq \mu\bar{\beta}.\langle g(t_+) \,\|\, \mu y.\langle [[y]] \,\|\, \bar{\beta}\rangle\rangle \\ &\simeq \mu\bar{\beta}.\langle t_+ \,\|\, \mu[[x]].\langle x \,\|\, \mu y.\langle [[y]] \,\|\, \bar{\beta}\rangle\rangle\rangle \\ &\simeq \mu\bar{\beta}.\langle t_+ \,\|\, \mu[[x]].\langle [[x]] \,\|\, \bar{\beta}\rangle\rangle \\ &\simeq \mu\bar{\beta}.\langle t_+ \,\|\, \bar{\beta}\rangle \\ &\simeq t_+ \end{aligned}$$

The two other cases (for V_+ and t_-) are no less straightforward. □

Remark 21. The derivation of $\neg\neg X \vdash \neg\neg X$ obtained the following way:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{X \vdash X}\,(\mathrm{ax})}{X, \neg X \vdash}\,(\neg\vdash)}{\neg X \vdash \neg X}\,(\vdash\neg)}{\neg X, \neg\neg X \vdash}\,(\neg\vdash)}{\neg\neg X \vdash \neg\neg X}\,(\vdash\neg)$$

is also equivalent to identity. This should be obvious since there is only one strongly focused derivation of $\neg\neg X \vdash \neg\neg X$ (whether X be positive or negative).

Since $\neg$ is the reification of $\cdot^{\perp}$, one should also expect the usual De Morgan isomorphisms:

$$\neg(A \otimes B) \simeq \neg A \parr \neg B$$
$$\neg(A \,\&\, B) \simeq \neg A \oplus \neg B$$
$$\vdots$$

A.3. A non-involutive negation: intuitionistic negation

Negation is usually defined in intuitionistic logic with $\neg A = A \to R$ for some "response" type R, where $\to$ is the implication from λ calculus. Yet both in CBV and CBN, implication hides a modality (as shown by Girard in [Gir87]). Thus we can define an analogue of this nega-

[8] We use the pattern convention of Section B.

tion in $\mathsf{LK}_{\mathrm{pol}}$, we write $\neg_i$ and we refer to as the *intuitionistic negation* since it is a legacy of the λ calculus, by forcing this negation not to change the polarity of formulae. (It indeed makes the modalities !, ? of LLP appear inside negation in the translation into the latter.)

Definition 22 ($\neg_i$)**.** The formula $\neg_i A$ is defined in function of the polarity of A as follows:

A	P	N
$\neg_i A$	$\neg^+ P$	$\neg^- N$

Now, since this negation does not change the polarity of formulae when they are exchanged between the left and the right of the sequents, it means that the corresponding constructors *does* change the polarity of terms. In fact, the change of polarity can be made explicit with a shift connective as follows: $\neg_i A \simeq \neg\updownarrow A$.[9] (The roles are however not symmetric, since $\neg_i\uparrow P \simeq \neg\downarrow\uparrow P \not\simeq \neg P$, and therefore $\neg A \not\simeq \neg_i\updownarrow A$.)

It is known that $\neg_i$ is not an involution — see e.g. [Lau02] (Section 14, "*Logique classique linéaire*") where Laurent shows that non-involutivity comes from the modality it hides, rather than from the structural rules of classical logic.

Also $\neg_i$ does not make the two polarities of connectives communicate. Therefore, when one takes $\neg_i$ as a negation, one is tempted to go one step further and add the constraint that change of polarities (at the level of constructors) may only occur within a negation. With this constraint, derived sequents only have one polarity of connectives. This therefore delineates two distinct and symmetric fragments of $\mathsf{LK}_{\mathrm{pol}}$:

1. LKQ, a classical logic with positive connectives only. This restriction gives the control to the code, and therefore corresponds to Curien-Herbelin's CBV $\bar{\lambda}\mu\tilde{\mu}_v$ calculus [CH00]. Syntactically it more closely resembles Wadler's dual calculus in CBV [Wad03] thanks to the use of conjunction, disjunction and negation instead of implication. Precisely, Wadler's connectives $\vee$ and $\neg$ correspond to the connective $\oplus$ and $\neg_i$, while & is given the rules of introduction of the *with* but corresponds in its reduction to the connective $\otimes$.

2. LKT, a classical logic with negative connectives only. Control is given to the environment, like in Curien-Herbelin's CBN $\bar{\lambda}\mu\tilde{\mu}_n$ calculus. Again it more closely resembles Wadler's dual calculus in CBN. Wadler's connectives & and $\neg$ correspond to the connectives & and $\neg_i$, while $\vee$ is given the rules of introduction of the *plus* but corresponds in its reduction to the connective $\parr$.

[9] For that matter, we find back the CBV and CBN negations, the direct-style counterparts of Zeilberger's $\overset{v}{\neg}$ and $\overset{n}{\neg}$ [Zei08], as the two particular cases of intuitionistic negation:

$$\overset{v}{\neg}P = \neg\uparrow P \simeq \neg_i P \qquad \overset{n}{\neg}N = \neg\downarrow N \simeq \neg_i N$$

We show for instance that Wadler's negation, be it in CBV or in CBN, corresponds to $\neg_i$. Let us take:

$$\langle \mathrm{not}(t)| \overset{\text{def}}{=} \begin{cases} \langle \mu[\alpha].\langle \alpha \parallel t\rangle \big| & \text{if } t \in \mathcal{T}_+ \\ \langle [t]| & \text{if } t \in \mathcal{T}_- \end{cases}$$

and symmetrically for $|\mathrm{not}(t)\rangle$. One then has:

$$\frac{\Gamma \mid t : A \vdash \Delta}{\Gamma \vdash \mathrm{not}(t) : \neg_i A \mid \Delta}\ (\vdash \neg_i)$$

$$\frac{\Gamma \vdash t : A \mid \Delta}{\Gamma \mid \mathrm{not}(t) : \neg_i A \vdash \Delta}\ (\neg_i \vdash)$$

and the following reduction rule:

$$\langle \mathrm{not}(t) \parallel \mathrm{not}(u)\rangle \to \langle u \parallel t\rangle$$

This matches the reduction rules of Wadler's negation $\neg A$ both in CBV (case A positive) and in CBN (case A negative).

While we're at it, let us clarify the role of the reduction $\to_\varsigma$ introduced by Wadler [Wad03] and present in $\mathsf{L}_{\mathrm{foc}}$. From the point of view of the linear analysis of classical logic, which constructors the latter over polarised linear logic and makes explicit the *stoup*, it might seem superfluous. For instance it is in some sense already present in Girard's LC (since the constraints of the *stoup* force one to explicitly introduce cuts for a conjunction). But $\to_\varsigma$ is useful from a categorical point of view, as it allows one to see a term depending on a variable as morphism in its variable. It allows for instance one to *state* that $[[t_+]]$ behaves the same as t_+.

A.4. Conclusion: From the duality to the *symmetry* of computation

In the seminal article of 1987 [Gir87], Girard decomposed CBN intuitionistic negation into the linear negation and the modality ! that allows re-use of the argument. We can conclude that his classical logic LC [Gir91] offered in fact an intermediate decomposition of intuitionistic negation, into a *classical* (involutive) negation and modalities (the shifts) that control the order of evaluation.

The historical term syntaxes for classical sequent calculus [CH00, Wad03] were based on the intuitionistic negation rather than the classical one, maybe because the latter is quite demanding, as it requires *both* polarities at the same time. As we have seen above, the distortion induced by the intuitionistic negation make CBV and CBN appear as two islands of determinism among a vast ocean of non-determinism. It shed light on the symmetry between CBN and CBV, known as the "*duality of computation*".

But this "*duality*" between CBN and CBV is the mere expression of the practical possibility of a symmetry between the code and the environment. Such a symmetry is always possible theoretically, but it is not a dogma. More interesting about the "*duality*" between CBV and CBN was the fact that it dealt with reduction strategies that were not given an equal esteem from the point of view of theory. Beyond putting CBV on a par with CBN, it in fact hinted at more theoretical explanations of the notions of *laziness*

and *eagerness* in computation.

B. Patterns

Invertible constructors ($⅋, \&, \forall$) or semi-invertible constructors (!) are given in $\mathsf{L}_{\mathrm{foc}}$ under the form of *pattern matchings*. It is possible to be very formal about the use of first-class patterns. Here we on the other hand mean to show that it is possible to avoid the bureaucracy a definition of patterns would require, and get them as an informal writing convention instead.

Definition 23 (Pattern convention)**.** We use the notation $\mu q.c$, with q a binding pattern, to shorten successive μ binders into a single one. It is defined in terms of the basic μ binders by introducing as many intermediate variables of the proper polarity as needed.

The interest of a convention is that we do not need to give an exhaustive definition; instead we give some example shorthand writings. A term of type $(N_1 ⅋ N_2) \,\&\, M$ can for instance be given by:

$$\mu\left(\iota_1(x,y).c_1 \,\middle|\, \iota_2(x).c_2\right)$$

which is defined with:

$$\mu\left(\iota_1(z).\langle z \,\|\, \mu(x,y).c_1\rangle \,\middle|\, \iota_2(x).c_2\right)$$

Or a term of type $\neg P ⅋ Q$ can be given by:

$$\mu([x],\bar{\alpha}).c$$

which is defined with:

$$\mu(\bar{y},\bar{\alpha}).\langle \mu[x].c \,\|\, \bar{y}\rangle$$

Now the introduction of the lazy pair, that is to say the rule ($\vdash \&$), is written in such a pattern-matching, which is unusual for a cartesian product. The lazy pair of two terms is then obtained as follows:

$$\langle (t|t')\rangle \stackrel{\text{def}}{=} \left\langle \mu\left(\iota_1(\bar{\kappa}).\langle t \,\|\, \bar{\kappa}\rangle \,\middle|\, \iota_2(\bar{\kappa}).\langle t' \,\|\, \bar{\kappa}\rangle\right)\right|$$

While we hope the cost of this longer connective is offset by the above writing convention, there is a more semantic reason for this choice. Pattern-matching makes in general η-rules (also known as invertibility) more suggestive and easier on the eyes:

$$\mu\left(\iota_1(\bar{x}).\langle t_- \,\|\, \iota_1(\bar{x})\rangle \,\middle|\, \iota_2(\bar{y}).\langle t_- \,\|\, \iota_2(\bar{y})\rangle\right) \to_\eta t_-$$
$$\mu(x,y).\langle (x,y) \,\|\, t_-\rangle \to_\eta t_-$$

etc.

On the other hand, it is legitimate to want first-class patterns, so as to be able to give a meaning to derivations like:

$$\frac{\vdash t : A \mid \Gamma, x : N, y : M}{\vdash t : A \mid \Gamma, (x,y) : N ⅋ M}\ (⅋)$$

without resorting to "writing conventions". The relationship between sequent calculus and pattern-matching has long been studied and the purpose of this paper is not to make a survey of it. But the reader can refer to Curien and the author's [CMM10], which shows that our notation $\mu q.c$ can be made formal: in fact q is not a "binding pattern" but a *counter-pattern* that filters patterns during computation.

C. Units

We show that the units can trivially be added to the syntax. We shall also discuss our terminology for the constant tp. It is sufficient for this purpose to follow the one-side tradition.

The neutral elements of the connectives $\otimes, ⅋, \&, \oplus$ are respectively $\mathbf{1}, \bot, \top$ and $\mathbf{0}$. They can be added as follows in the syntax:

$$t_+ ::= \cdots \mid ()$$
$$t_- ::= \cdots \mid \mu().c \mid \mathsf{tp}$$

There is an obvious new reduction rule:

$$\langle () \,\|\, \mu().c\rangle \to c$$

There is no constant for $\mathbf{0}$, hence no reduction rule for the pair $\mathbf{0}/\top$.

Typing rules are as follows:

$$\frac{}{\vdash () : \mathbf{1} \mid}\ (\mathbf{1}) \qquad \frac{c : (\vdash \Gamma)}{\vdash \mu().c : \bot \mid \Gamma}\ (\bot)$$

$$\frac{}{\vdash \mathsf{tp} : \top \mid \Gamma}\ (\top) \qquad \text{(no rule for } \mathbf{0}\text{)}$$

We see the tp constant as a pattern matching with no branch. (This interpretation is consistent with the counterpart of the $\top$ rule in Ludics [Gir07], the *skunk*.)

Now, in the $\lambda\mu$-top calculus (a variant of the $\lambda\mu$ calculus), Ariola and Herbelin [AH03] use a constant of continuation called the *toplevel* to interpret a logical rule of elimination of absurdity related to the *Ex Falso Quodlibet* (EFQ) law. This is where the terminology tp comes from. Indeed, the toplevel in CBV can be seen as a pattern-matching with no branch, and can therefore be denoted by $|\mathsf{tp}\rangle$. As a result of this interpretation, we find back the encoding of Felleisen's *abort* operator in the $\lambda\mu$-top calculus [AH03], in a derivation that only makes use of the ($\top$) rule and the identity rules. Indeed, take:

$$\mathcal{A}(t_+) \stackrel{\text{def}}{=} \mu\_.\langle t_+ \,\|\, \mathsf{tp}\rangle$$

Then the following derivation is a direct consequence of the ($\mathbf{0}\vdash$) rule:

$$\frac{\Gamma \vdash t_+ : \mathbf{0} \mid \Delta}{\Gamma \vdash \mathcal{A}(t_+) : A \mid \Delta}\ (\mathrm{EFQ})$$

Symmetrically, $\langle \mathsf{tp}|$ can be interpreted as code that halts the computation. This is computationally correct: it is similar to the null pointer, in the sense that it does not lead to crashes until it is used, since it is negative. It is logically correct as well, since the rules of logic ensures that it

won't be used, since there is no way to introduce a value of the dual type $\mathbf{0}$.

D. Internal completeness

This section follows the tradition of one-sided sequents.

Classical realisability admits here a notion similar to the internal completeness of the connectives of Ludics [Gir07].

Remark 24 (Daimon). The following additional rule passes adequacy:

$$\frac{}{c_0 : (\vdash \_ : A_1, \dots, \_ : A_n)}\ (\maltese)\ (\text{when } c_0 \in \perp\!\!\!\perp)$$

In this section, we shall consider that type systems include this rule. This is allowed by the fact that this addition preserves the adequacy lemma.

Proposition 25. *Let R a set of closed values of the same polarity. The following two properties are equivalent:*

1. $R^{\perp\perp}{}_{\mathbb{V}} = R$
2. *R is of the form $\mathbf{H}_{\mathbb{V}}$ with $\mathbf{H}$ a behaviour.*

Proof. (1⇒2) Trivial. (2⇒1) One always has $R \subseteq R^{\perp\perp}{}_{\mathbb{V}}$. Now suppose $R = \mathbf{H}_{\mathbb{V}}$ with $\mathbf{H}$ a behaviour. Since $\mathbf{H}_{\mathbb{V}} \subseteq \mathbf{H}$ and $\mathbf{H}$ is a behaviour, one has $\mathbf{H}_{\mathbb{V}}{}^{\perp\perp} \subseteq \mathbf{H}$, hence $\mathbf{H}_{\mathbb{V}}{}^{\perp\perp}{}_{\mathbb{V}} \subseteq \mathbf{H}_{\mathbb{V}}$. □

Definition 26. A set R of closed values of the same polarity is *complete* if one of the above two equivalent properties hold.

Proposition 27. *Let R and S be complete sets. The following properties hold:*

1. $\left(R^{\perp\perp} \times S^{\perp\perp}\right)^{\perp\perp} = (R \times S)^{\perp\perp}$;
2. $\left(R^{\perp\perp} + S^{\perp\perp}\right)^{\perp\perp} = (R + S)^{\perp\perp}$;
3. $\downarrow(R^{\perp\perp}) = (\downarrow R)^{\perp\perp}$.

Proof. It is a rephrasing of lemma 38. □

Definition 28. A command of the form $\langle V_+ \,\|\, W \rangle$ where V_+ is a constructor among $\otimes, \oplus, \exists, !$ and W a constructor among $⅋, \&, \forall$ or of the form $\backprime(V)$ is *ill-formed* either if these constructors are not the dual of each other, or if the polarities of the sub-terms do not match the polarities of the variables bound in the counter-term.

An ill-formed command therefore never reduces and is never typable, unless with the ✠ rule, provided that the observation includes ill-formed commands.

Proposition 29 (Internal completeness)**.** *Let $\perp\!\!\!\perp$ be:*

1. *a non-empty observation,*
2. *that does not contain ill-formed commands,*
3. *that is closed under $\rightarrow_\beta$ reduction.*

One then has:

- *$\emptyset$ (as a set of positive terms) and $\{()\}$ are complete;*
- *For all R and S complete, the sets $R \times S$, $R + S$ and $\downarrow R$ are complete.*

Proof. Let $\perp\!\!\!\perp$ be such an observation. In the following, $\vdash$ denotes the typability in $\mathsf{LK}_{\mathrm{pol}}$ with daimon. One takes c_0 an element of $\perp\!\!\!\perp$. Case $\emptyset$: let $V \in \emptyset^{\perp\perp}{}_{\mathbb{V}}$. One has for each $t \in \mathcal{T}^0_-$, $\langle V \,\|\, t \rangle \in \perp\!\!\!\perp$. This is impossible, since among these commands some are ill-formed. Hence $\emptyset^{\perp\perp}{}_{\mathbb{V}} = \emptyset$. Case $\{()\}$: let $V \in \{()\}^{\perp\perp}{}_{\mathbb{V}}$. One has $\vdash \mu().c_0 : \perp$, hence $\langle V \,\|\, \mu().c_0 \rangle \in \perp\!\!\!\perp$ by adequacy. The latter is therefore not ill-formed and one has $V = ()$ as expected. (Notice the importance of the absence of free variables here and in the rest of the proof.) Let R and S complete. Case $R \times S$: let $V \in (R \times S)^{\perp\perp}{}_{\mathbb{V}}$. One has $\vdash \mu(\_,\_).c_0 : R^\perp ⅋ S^\perp$, hence $\langle V \,\|\, \mu(\_,\_).c_0 \rangle \in \perp\!\!\!\perp$ and the latter is therefore not ill-formed. One has therefore V of the form (V_1, V_2) with V_1 and V_2 respectively of the same polarities as R and S (no ill-formed command and V is closed). Then let $t \in R^\perp$. One has $\vdash \mu(\kappa,\_).\langle \kappa \,\|\, t \rangle : R^\perp ⅋ S^\perp$. Hence $\langle (V_1, V_2) \,\|\, \mu(\kappa,\_).\langle \kappa \,\|\, t \rangle \rangle \in \perp\!\!\!\perp$. Since $\perp\!\!\!\perp$ is closed under $\rightarrow_\beta$ reduction, one has $\langle V_1 \,\|\, t \rangle \in \perp\!\!\!\perp$. Hence $V_1 \in R^{\perp\perp}$. Hence $V_1 \in R$ by completeness of R. Similarly one shows $V_2 \in S$, hence $V \in R \times S$. Case $R + S$: Let $V \in (R + S)^{\perp\perp}{}_{\mathbb{V}}$. One has $\vdash \mu\left(\iota_1(\_).c_0 \,\middle|\, \iota_2(\_).c_0\right) : R^\perp \,\&\, S^\perp$, therefore $\left\langle V \,\middle\|\, \mu\left(\iota_1(\_).c_0 \,\middle|\, \iota_2(\_).c_0\right) \right\rangle \in \perp\!\!\!\perp$ and V is of the form $\iota_i(V')$ with $i = 1$ and V' of the same polarity as R or with $i = 2$ and V' of the same polarity as S. Suppose for instance $i = 1$ and let $t \in R^\perp$. One has $\vdash \mu\left(\iota_1(\kappa).\langle \kappa \,\|\, t \rangle \,\middle|\, \iota_2(\_).c_0\right) : R^\perp \,\&\, S^\perp$ and therefore $\left\langle V \,\middle\|\, \mu\left(\iota_1(\kappa).\langle \kappa \,\|\, t \rangle \,\middle|\, \iota_2(\_).c_0\right) \right\rangle \in \perp\!\!\!\perp$. Hence $\langle V' \,\|\, t \rangle \in \perp\!\!\!\perp$ Hence $V' \in R^{\perp\perp}$, whence $V' \in R$ by completeness of R. Completeness of $\downarrow R$ is proved in a similar way. □

It is easy to find *ad hoc* counter-examples to the above property in case of an observation that does not follows the conditions 1, 2 and 3. The constraints we have on the observation in order to get internal completeness define a very broad class of observations. For instance, all the examples we gave in this paper enjoy these properties. Also, the hypotheses have counterparts in Ludics:

1. Having a non-empty observation implies the existence of a daimon.
2. There's no equivalent to ill-formedness in Ludics.
3. The orthogonality of Ludics enjoys closure under reduction.

We mention another crucial property for the result: the absence of free variables in the commands. Like Ludics, Classical Realizability has found a way to get rid of variables: we only study closed terms.

We do not claim however that the set of constraints on the observation is minimal to get internal completeness.

Corollary 30. *For R and S complete and when $\perp\!\!\!\perp$ meets the requirements of the previous proposition, one has $|R \otimes S|_{\mathbb{V}} = R \times S$. In particular, $|A \otimes B|_{\mathbb{V}} = |A|_{\mathbb{V}} \times |B|_{\mathbb{V}}$. (And similarly for constructors $+$ and $\downarrow$.)*

Proof. The first equality is a consequence of the two previous propositions. The second equality follows directly. □

Applications

There are behaviours not generated by their values; more generally:

Proposition 31. *An intersection of behaviours generated by their values is not generated by its values in general.*

Proof. Here is a counter-example. It is sufficient to take some observation $\perp\!\!\!\perp$ that is non-empty, closed under $\to$ reduction and that contains no ill-formed command. Take $\mathbf{1} = \mathcal{T}^0_+ \cap \mathbb{V}$ and $\mathbf{0} = \emptyset$.[10] Take $|\bot| = |\mathbf{1}|^\perp$ and $|\top| = |\mathbf{0}|^\perp$. Take $\mathbf{H} = |\mathbf{1} \oplus \top| \cap |\mathbf{0} \oplus \bot|$. One has $\mathbf{1}$ and $\mathbf{0}$ complete (trivial for $\mathbf{1}$, follows from the "no ill-formed command" constraint for $\mathbf{0}$). The requirements of proposition 29 on completeness are met, hence one has $\mathbf{H}_\mathbb{V} = \mathbf{0} + \mathbf{1}^\perp$. Now take $t = \mu\alpha.\left\langle \iota_2\left(\mu x.\langle \iota_1(x) \,\|\, \alpha\rangle\right) \,\middle\|\, \alpha\right\rangle$. From $\vdash_{\mathsf{LK}_{\mathrm{pol}}} t : X \oplus X^\perp$ one concludes by adequacy $t \Vdash \mathbf{1} \oplus \top$ and $t \Vdash \mathbf{0} \oplus \bot$. Hence $t \in \mathbf{H}$. Take also $c_0 \in \mathcal{C}^0 \setminus \perp\!\!\!\perp$, $V_0 \in \mathbf{1}$ and $u = \mu\left(\iota_1(\_).c_0 \,\middle|\, \iota_2(\alpha).\langle V_0 \,\|\, \alpha\rangle\right)$. One has $u \in \mathbf{H}_\mathbb{V}{}^\perp$. But $\langle t \,\|\, u\rangle \to^* c_0 \notin \perp\!\!\!\perp$, hence $t \notin \mathbf{H}_\mathbb{V}{}^{\perp\perp}$ since $\perp\!\!\!\perp$ is closed under $\to$. □

Shocking equalities In Section 4 we defined a notion of polymorphism based on the value restriction, and claimed it enjoyed "shocking equalities" in the sense of Girard [Gir07] for a broad range of observations. Here's a proof of it:

Proposition 32. *Let $\perp\!\!\!\perp$ be a non-empty observation, closed under $\to_\beta$ reduction and that does not contain ill-formed terms. Then the following equalities hold:*

$$|\mathsf{A}X\,(A \oplus B)| = |(\mathsf{A}X\,A) \oplus (\mathsf{A}X\,B)|$$
$$|\mathsf{A}X\,(A \otimes B)| = |(\mathsf{A}X\,A) \otimes (\mathsf{A}X\,B)|$$

Proof. The constraints for internal completeness are met. Therefore, according to Corollary 30:

$$\begin{aligned}
&\bigcap_{R\in\Pi} |(A \oplus B)\,[R/X]|_\mathbb{V} \\
=&\bigcap_{R\in\Pi} \left(|A\,[R/X]|_\mathbb{V} + |B\,[R/X]|_\mathbb{V}\right) \\
=&\bigcap_{R\in\Pi} |A\,[R/X]|_\mathbb{V} + \bigcap_{R\in\Pi} |B\,[R/X]|_\mathbb{V}
\end{aligned}$$

Whence the result for $\oplus$. Same reasoning for $\otimes$. □

E. CBV and CBN λ calculus in $\mathsf{L}_{\mathbf{foc}}$

We implemented in Section 2 the implication in the one-sided setting. We go back on this example in the two-sided setting, because it gives a more accurate and natural description for an implication. It should be stressed that the "translations" in $\mathsf{LK}_{\mathrm{pol}}$ we give are local definitions, i.e. syntactic sugar.

[10] In case the syntax do not have the constructors () and $\mu().c$, it is always possible to define the type $\mathbf{1}$ by replacing the latter constructors respectively by any $V \in \mathcal{T}^0_{+\,\mathbb{V}}$ and $\mu x.c$, $x \notin \mathcal{FV}(c)$. This allows us to give a proof that does not rely on units.

We write here v the code and e the environment as in [CH00]. Take:

$$\begin{aligned}
A \to B &\stackrel{\text{def}}{=} \neg^- A \parr B \\
\langle \lambda\kappa.v| &\stackrel{\text{def}}{=} \langle \mu([\kappa], \bar\kappa').\langle v \,\|\, \bar\kappa'\rangle \,| \qquad (\bar\kappa' \notin \mathcal{FV}(v)) \\
|v \cdot e\rangle &\stackrel{\text{def}}{=} |([v], e)\rangle \\
\langle v\, v'| &\stackrel{\text{def}}{=} \langle \mu\bar\kappa.\langle v \,\|\, v' \cdot \bar\kappa\rangle \,|
\end{aligned}$$

Note that $\neg^- A \parr B$ is isomorphic to $\neg A \parr B$, but we chose the former because it yields a more concise formulation in our setting. (They are equal if A is positive, and if A is negative then using $\neg^-$ instead of $\neg$ introduce a shift that would be there anyway because of the $\parr$.)

One has:

$$\frac{\Gamma, \kappa : A \vdash v : B \mid \Delta}{\Gamma \vdash \lambda\kappa.v : A \to B \mid \Delta}\ (\vdash\to)$$

$$\frac{\Gamma \vdash v : A \mid \Delta \qquad \Gamma' \mid e : B \vdash \Delta'}{\Gamma, \Gamma' \mid v \cdot e : A \to B \vdash \Delta, \Delta'}\ (\to\vdash)$$

$$\frac{\Gamma \vdash v : A \to B \mid \Delta \qquad \Gamma' \vdash v' : A \mid \Delta'}{\Gamma, \Gamma' \vdash v\, v' : B \mid \Delta, \Delta'}\ (\text{app})$$

We study two particular cases for $A \to B$:

Case $N \to M$. This corresponds to call-by-name. One has, for a positive value E:

$$\langle v\, v' \,\|\, E\rangle \to \langle v \,\|\, v' \cdot E\rangle$$
$$\langle \lambda\alpha.v \,\|\, v' \cdot E\rangle \to^2 \langle v\,[v'/\alpha] \,\|\, E\rangle$$

These are the rules of reduction of a Krivine weak head reduction machine [Kri04], whose *stacks* are environment-values; or again the rules of the $\bar\lambda\mu\tilde\mu$ calculus in call-by-name [CH00].

Case $P \to Q$. One would expect this to correspond to call-by-value. One has:

$$\langle v\, v' \,\|\, e\rangle \to \langle v \,\|\, v' \cdot e\rangle$$
$$\langle \lambda x.v \,\|\, v' \cdot e\rangle \to'^6 \langle v' \,\|\, \mu x.\langle v \,\|\, E\rangle\rangle$$

(Where $\to'$ is $\to$ extended to sub-commands.)

This looks like the rules of the CBV $\bar\lambda\mu\tilde\mu_v$ calculus, and $P \to Q$ can be seen as some form of call-by-value, since its argument is "called by value". (In fact, any $P \to A$ could.)

But this is not sufficient to translate the $\bar\lambda\mu\tilde\mu_v$ calculus in $\mathsf{L}_{\mathrm{foc}}$ (assuming that we are looking for a "syntactic sugar" kind of definition). Indeed, in $\bar\lambda\mu\tilde\mu_v$, $\langle \mu\_.c_1 \,\|\, \tilde\mu\_.c_2\rangle$ reduces to c_1, while if the principal type of the cut is implication, it would reduce in $\mathsf{L}_{\mathrm{foc}}$ to c_2 with our above negative definition of implication.

More practically, a positive $\lambda x.v$, i.e. an eager type for implication, is needed by common kinds of programming practises, such as the following definition of a mutable variable which is static to a function in OCaml:

```
let f =
  let r = ref []
  in function -> ...
```

which asks for an eager evaluation of the abstraction.

The case of CBV Curien-Herbelin's $\bar{\lambda}\mu\tilde{\mu}_v$ is retrieved by forcing the positive polarity with a dummy positive constructor $\downarrow$ as follows: $\downarrow(\neg P \parr Q)$.[11] The connective $\downarrow$ can for instance be implemented by an unary tensor, and has already been used in Section 2 to give a polarity to the quantifications.

Let us trivially extend the type system with the positive shift connective:

$$\frac{\Gamma \vdash t : A \mid \Delta}{\Gamma \vdash \{t\} : \downarrow A \mid \Delta}\ (\vdash\downarrow)$$

$$\frac{c : (\Gamma, \kappa : A \vdash \Delta)}{\Gamma \mid \mu\{\kappa\}.c : \downarrow A \vdash \Delta}\ (\downarrow\vdash)$$

Now take:

$$\begin{aligned}
P \to Q &\stackrel{\text{def}}{=} \downarrow(\neg P \parr Q)\\
\langle \lambda x.v \mid &\stackrel{\text{def}}{=} \langle \{\mu([x], \bar{\alpha}).\langle v \parallel \bar{\alpha}\rangle\} \mid && (\alpha \in \mathcal{FV}(v))\\
\mid v \cdot e\rangle &\stackrel{\text{def}}{=} \langle \mu\{\alpha\}.\langle \alpha \parallel ([v], e)\rangle \mid && (\alpha \in \mathcal{FV}(v, e))\\
\langle v\, v' \mid &\stackrel{\text{def}}{=} \langle \mu\bar{\alpha}.\langle v \,\|\, v' \cdot \bar{\alpha}\rangle \mid && (\alpha \notin \mathcal{FV}(v, v'))
\end{aligned}$$

We have the same typing rules for implication as before, and we now have the following rules of reduction:

$$\begin{aligned}
\langle v\, v' \,\|\, e\rangle &\to \langle v \,\|\, v' \cdot e\rangle\\
\langle \lambda x.v \,\|\, v' \cdot e\rangle &\to_{\prime 7} \langle v' \,\|\, \mu x.\langle v \parallel e\rangle\rangle\\
\langle V \,\|\, \mu x.c\rangle &\to c\,[V/x]
\end{aligned}$$

In addition, $\lambda x.v$ is now a positive value. It is therefore possible to translate the $\bar{\lambda}\mu\tilde{\mu}_v$ calculus by sending terms on positive terms and co-terms on negative terms, using the above implication.

Notice that the function is evaluated before its argument. Using the same encoding of implication, it is possible to implement the application $v\, v'$ such that the evaluation of the argument comes before that of the function.

These encodings of the CBV and the CBN implications have been studied by Laurent [Lau02] in LLP.

F. Details on the difference with the original formulation of LC

L_{foc} is a literal quotient[12] for LC; here are more details about this point. We choose to work with the one-sided convention for simplicity.

Let us remind that LC is based on the following decomposition of $\wedge$ and $\vee$ in function of the polarities:

$\wedge$	$+$	$-$
$+$	$P \otimes Q$	$N \otimes P$
$-$	$P \otimes N$	$N \,\&\, M$

$\vee$	$+$	$-$
$+$	$P \oplus Q$	$N \parr P$
$-$	$P \parr N$	$N \parr M$

[11] One can also remain in the realm of LKQ by taking the isomorphic formula for implication: $\neg_i(P \otimes \neg_i Q)$.

[12] that is, a term syntax which is a quotient, in our case for the structural rules.

In order to emphasise the fact that L is a syntax for LC, we rephrase LK_{pol} with a stoup. One will deduce LC by encoding $\wedge$ and $\vee$ as shown above.

LK_{pol} with a stoup

Let a distinguished negative variable that shall be written $\star$, with the particularity that it cannot be subject to weakenings or contractions. We shall write Π a context that is either empty or of the form $\star : P$. We shall suppose that contexts $\Gamma, \Delta \ldots$ do not contain $\star$. The judgements of LK_{pol} with a stoup are of the form:

$$\begin{aligned}
&\vdash t_+ : P; \Gamma\\
&\vdash t_- : N; \Gamma, \Pi\\
&\vdash t_+ : P \mid \Gamma, \Pi\\
c &: (\vdash \Gamma, \Pi)
\end{aligned}$$

The stoup of a sequent is either $t_+ : P$ in $\vdash t_+ : P; \Gamma$, or $\star : P$ when is appears in the sequent. Notice the semicolon for negative terms: this is a convention that simplifies the rules of inference and corresponds to our choice of declaring "value" any negative term.

LK_{pol} is formulated with a stoup in fig. 5. As is, derivable sequents of the form $\vdash t : A; \Gamma, \Pi$ enjoy that t is a value.

Formulated like this, LK_{pol} therefore has its positive constructors restricted to values. But if one wants to be comfortable, generic positives $(t, u), \iota_i(t) \ldots$ can be redefined. For instance, one can take:

$$(t, u) \stackrel{\text{def}}{=} \mu\alpha.\left\langle t \,\middle\|\, \mu x.\left\langle u \,\middle\|\, \mu y.\langle (x, y) \,\middle\|\, \alpha\rangle\right\rangle\right\rangle$$

which allows the derivation:

$$\frac{\vdash t : A \mid \Gamma \qquad \vdash u : B \mid \Delta}{\vdash (t, u) : A \otimes B \mid \Gamma, \Delta}\ (\otimes)$$

This is where the arbitrary choice between the left and the right of the pair is made. This definition justifies the rule $\to_\varsigma$ for the eager pair, and it goes the same with other connectives.

Central terms

The description of LC would not be complete without the following additional rules for handling the stoup:

$$\frac{\vdash t_+ : P; \Gamma \qquad \vdash t_- : P^\perp; \Delta, \Pi}{\langle t_+ \,\|\, t_-\rangle : (\vdash \Gamma, \Delta, \Pi)}\ (P\text{-cut})$$

$$\frac{c : (\vdash \Gamma, \star : P)}{\vdash \mu\star.c : P; \Gamma}\ (\mu\star)$$

These rules allows to derive no additional term (the rule $\mu\star$ is redundant with the rule μ). However, they extend the subset of positive values into a subset of *central* terms (terms t_+ typable in $\vdash t_+ : P; \Gamma$), that behave like values without necessarily being values. For such a term t_+, the reduction:

$$\langle t_+ \,\|\, \mu x.c\rangle \to_\prime c\,[t_+/x] \qquad (1)$$

$\mathsf{LK}_{\mathrm{pol}}$ (one-sided, with explicit stoup)

Identity

$$\frac{}{\vdash x : P; x : P^{\perp}}\,(ax+) \qquad \frac{}{\vdash \alpha : N; \alpha : N^{\perp}}\,(ax-)$$

$$\frac{c : (\vdash x : N, \Gamma, \Pi)}{\vdash \mu x.c : N; \Gamma, \Pi}\,(\mu) \qquad \frac{c : (\vdash \alpha : P, \Gamma, \Pi)}{\vdash \mu\alpha.c : P \mid \Gamma, \Pi}\,(\mu) \text{ (possibly with } \alpha = \star)$$

$$\frac{\vdash t_- : N; \Gamma \qquad \vdash t_+ : N^{\perp} \mid \Delta, \Pi}{\langle t_+ \,\|\, t_- \rangle : (\vdash \Gamma, \Delta, \Pi)}\,(N\text{-cut}) \qquad \frac{\vdash t_+ : P; \Gamma}{\vdash t_+ : P \mid \Gamma}\,(\mathrm{der})$$

Logic

$$\frac{\vdash t : A; \Gamma \qquad \vdash u : B; \Delta}{\vdash (t,u) : A \otimes B; \Gamma, \Delta}\,(\otimes) \qquad \frac{c : (\vdash \kappa : A, \kappa' : B, \Gamma, \Pi)}{\vdash \mu(\kappa, \kappa').c : A \parr B; \Gamma, \Pi}\,(\parr)$$

$$\frac{c : (\vdash \kappa : A, \Gamma, \Pi) \qquad c' : (\vdash \kappa' : B, \Gamma, \Pi)}{\vdash \mu\left(\iota_1(\kappa).c \,\middle|\, \iota_2(\kappa').c'\right) : A \,\&\, B; \Gamma, \Pi}\,(\&) \qquad \frac{\vdash t : A; \Gamma}{\vdash \iota_1(t) : A \oplus B; \Gamma}\,(\oplus_1) \qquad \frac{\vdash t : B; \Gamma}{\vdash \iota_2(t) : A \oplus B; \Gamma}\,(\oplus_2)$$

$$\frac{\vdash t : A[P/X]; \Gamma}{\vdash \{t\} : \exists X A; \Gamma}\,(\exists) \qquad \frac{c : (\vdash \kappa : A, \Gamma, \Pi)}{\vdash \mu\{\kappa\}.c : \forall X A; \Gamma, \Pi}\,(\forall)\ (X \notin \mathcal{FV}(\Gamma))$$

$$\frac{}{\vdash () : \mathbf{1};}\,(\mathbf{1}) \qquad \frac{c : (\vdash \Gamma, \Pi)}{\vdash \mu().c : \bot; \Gamma, \Pi}\,\bot$$

$$\frac{}{\vdash \mathsf{tp} : \top; \Gamma, \Pi}\,(\top) \qquad \textit{no rule for } \mathbf{0}$$

Structure Contractions, weakenings outside the stoup.

Figure 5: $\mathsf{LK}_{\mathrm{pol}}$ with explicit stoup.

has a meaning from the point of view of semantics.

An example of such a central term is $\mu\star.\langle V \,\|\, \mu\kappa.\langle V_+ \,\|\, \star\rangle\rangle$ which can be read "let $\kappa = V$ in V_+"; in this case the above reduction does not violate Church-Rosser. Another example of a central term, for which $\rightarrow'$ violates Church-Rosser, is $\mu\star.\langle \alpha \,\|\, (\star, \kappa'')\rangle$ which is one half of the dissociativity morphism. This justifies the rejection of $\rightarrow'$ as a reduction rule, even if we have to accept it as a kind of observational rule.

Last, let us notice that according to [Gir91] $\mu\_.\langle V_+ \,\|\, \mathsf{tp}\rangle$, which reads "$\mathcal{A}(V_+)$", would be central.

G. Neutral Atoms

Being a term syntax, $\mathsf{L}_{\mathrm{foc}}$ is essentially a quotient for the structural rules of logic, which concurs to its aim of being readable and writeable. Proof nets have a less natural treatment of contraction and weakening, but offer a quotient on the identity rules. This gives proof nets and term syntaxes such as $\mathsf{L}_{\mathrm{foc}}$ complementary roles.

$\mathsf{L}_{\mathrm{foc}}$ is the only quotient of linear logic that at the same time accounts for full LL, including the additives, and remains simple. (Proof nets, with a different purpose, indeed have a less modest treatment of the additives). The only difference with LL is the fact that atoms have a polarity. For instance, proving $\forall X(X \multimap X)$ in LL amounts to proving both $\forall X(X \multimap X)$ and $\forall X(X^{\perp} \multimap X^{\perp})$ in our system.

Since each non-atomic formula has a polarity, this is a natural constraint that simplifies the term syntax. Let us make the difference of expressivity more precise: we will see that the difference in terms of provability is small compared to the bureaucracy that would be required in order to lift this constraint of polarity on the atoms.

Suppose we want to extend our system with "neutral" atoms we shall write $X^?$, that are not subject to such a constraint of polarity. $X^?$ could therefore be replaced either by a positive or by a negative formula.

Obtaining a derivation that contains such a neutral atom is equivalent to obtaining the derivations that correspond to each choice of a polarity for this atom. Such a derivation can therefore be seen as the superposition of two proofs. In fact, we can assume that these two proofs only differ where the axiom rule for $X^?$ appear, the rest of the proof being the same.

In particular, the axiom rule for $X^?$ itself can be seen as the superposition of the two axiom rules: it could be written $\vdash \kappa : X^? \mid \kappa : X^{?\perp}$, standing for $\vdash x : X \mid x : X^{\perp}$ if we eventually decide to replace $X^?$ by a positive formula, or standing for $\vdash \alpha : X^{\perp} \mid \alpha : X$ if we eventually decide to

replace $X^?$ by a negative formula. This means that with the convention according to which $\kappa, \kappa', \ldots$ denote variables that can either become positive or negative, a unique term represents this superposition of proofs.

For instance, a proof of $\forall X(X^? \multimap X^?)$ will be obtained when both $\forall X(X \multimap X)$ and $\forall X(X^\perp \multimap X^\perp)$ are proved. Proofs of these two formulae are respectively $\mu\{(x,\alpha)\}.\langle x \parallel \alpha\rangle$ and $\mu\{(\alpha,x)\}.\langle \alpha \parallel x\rangle$. The corresponding proof of $\forall X(X^? \multimap X^?)$ would therefore be written $\mu\{(\kappa,\kappa')\}.\langle \kappa \,\big\|\, \kappa'\rangle$.

The fact that all these proofs are identical up to the axiom rules also shows that given a proof of A, we can deduce proofs of all the B such that B is obtained from A by changing the polarity of some atoms. Giving neutral variables a formal status in the syntax would therefore solve the question of the polarity of atoms. The additional bureaucracy neutral variable would require would however obfuscate the presentation without a strong motivation. This is why we did not introduce them.

H. Detailed proofs

We give detailed proofs for the results given in the main sections.

H.1. Subject Reduction

Proposition 33. *Let $c \to c'$. If c is typable in* LL *(respectively* LK_{pol}*) in a context Γ, then c' is typable in* LL *(resp.* LK_{pol}*) in the context Γ.*

Proof. There is more bureaucracy than difficulty in this result. We shall avoid bureaucracy here, for instance regarding the structural rules, and will therefore only give an intuitive proof. This will be sufficient to explain why the focalising reduction strategy yields subject reduction for LL. We shall only treat LL since it is less liberal than $\mathsf{LK}_{\mathrm{pol}}$. By case on the origin of $c \to c'$.

1. Case $\langle \mu\kappa.c \,\big\|\, V\rangle \to_\mu c[V/\kappa]$. If κ is not subject to contractions or weakenings in c, then we can make the linear substitution of axiom rules for κ in the derivation of c by the derivation of V, since the context of V in turn won't be subject to contraction or weakenings. Otherwise, it means κ is of type $?A$ in Γ, and therefore V is of type $!A^\perp$. It means V, which is a value by hypothesis, either comes from an axiom rule or from a promotion. In both cases, the context of V is of the form $?\Delta$. This allows the substitution of the axiom rules for κ in the derivation of c, structural rules on $?\Delta$ replacing those on $?A$.

2. Case $c \to_\beta c'$. A local transformation of the proof reduces the situation to the previous case. For instance, in the case $c = \langle (V,V') \,\big\|\, \mu(\kappa,\kappa').c_0\rangle$, one has $\langle V \,\big\|\, \mu\kappa.\langle V' \,\big\|\, \mu\kappa'.c_0\rangle\rangle \to^2 c'$. The left hand side is typable in Γ, and the previous case shall be applied twice.

   In the special case of $c = \langle \{V\} \,\big\|\, \mu\{\kappa\}.c_0\rangle$ (a cut between $\exists X\, A$ and $\forall X\, A^\perp$), by hypothesis there exists a formula P such that V is of type $A[P/X]$. Replacing X by P in the derivation of $\mu\kappa.c_0$ gives it the type $A^\perp[P/X]$ and we are reduced to the case $\langle V \,\big\|\, \mu\kappa.c_0\rangle$.

3. Case $c \to_\varsigma c'$: straightforward.

□

H.2. Generation lemma

Lemma 34 (Generation)**.** *If A is a closed formula, then $|A|$ is generated by the set of its values.*

The proof requires the lemmas that follow.

Lemma 35. *Let T be a subset of $\mathcal{T}_+^0$ (resp. $\mathcal{T}_-^0$) and let $c \in \mathcal{C}$ with $\mathcal{FV}(c) = \{\kappa\}$ where κ is a positive (resp. negative) variable. If for each $V \in T_{\mathbb{V}}$ one has $c[V/\kappa] \in \perp\!\!\!\perp$, then $\mu\kappa.c \in T_{\mathbb{V}}{}^\perp$.*

Proof. Follows from saturation of $\perp\!\!\!\perp$ since $\langle V \,\big\|\, \mu\kappa.c\rangle \to_\mu c[V/\kappa] \in \perp\!\!\!\perp$. □

Lemma 36. *Whenever $U \subseteq U' \subseteq \mathbf{H}$ and U generates $\mathbf{H}$, then U' generates $\mathbf{H}$. In particular, if a behaviour is generated by a set of values, then it is generated by the set of its values.*

Proof. Straightforward. □

Lemma 37. *If $\mathbf{H}$ is a behaviour, then ${\mathbf{H}_{\mathbb{V}}}^{\perp\perp}{}_{\mathbb{V}} = \mathbf{H}_{\mathbb{V}}$.*

Proof. ($\supseteq$) Trivial. ($\subseteq$) Since $\mathbf{H}$ is a behaviour and $\mathbf{H}_{\mathbb{V}} \subseteq \mathbf{H}$, one has ${\mathbf{H}_{\mathbb{V}}}^{\perp\perp} \subseteq \mathbf{H}$, hence the result.

Lemma 38. *Let $\mathbf{H}$ and $\mathbf{G}$ be two behaviours. The following properties hold:*

1. ${\mathbf{H}_{\mathbb{V}}}^{\perp\perp} \times {\mathbf{G}_{\mathbb{V}}}^{\perp\perp} \subseteq (\mathbf{H} \times \mathbf{G})_{\mathbb{V}}{}^{\perp\perp}$;
2. ${\mathbf{H}_{\mathbb{V}}}^{\perp\perp} + {\mathbf{G}_{\mathbb{V}}}^{\perp\perp} \subseteq (\mathbf{H} + \mathbf{G})_{\mathbb{V}}{}^{\perp\perp}$;
3. $\downarrow({\mathbf{H}_{\mathbb{V}}}^{\perp\perp}) \subseteq (\downarrow\mathbf{H}_{\mathbb{V}})^{\perp\perp}$.

(1) Let $t \in {\mathbf{H}_{\mathbb{V}}}^{\perp\perp}$ and $u \in {\mathbf{G}_{\mathbb{V}}}^{\perp\perp}$; let $v \in (\mathbf{H} \times \mathbf{G})_{\mathbb{V}}{}^\perp$. If $t,u \in \mathbb{V}$, then $(t,u) \in \mathbf{H}_{\mathbb{V}} \times \mathbf{G}_{\mathbb{V}}$ by lemma 37. Yet, by definition, $(\mathbf{H} \times \mathbf{G})_{\mathbb{V}} = \mathbf{H}_{\mathbb{V}} \times \mathbf{G}_{\mathbb{V}}$, hence $\langle (t,u) \parallel v\rangle \in \perp\!\!\!\perp$. Otherwise, one has:

$$\langle (t,u) \parallel v\rangle \to \left\langle t \,\middle\|\, \mu\kappa.\left\langle u \,\middle\|\, \mu\kappa'.\langle (\kappa,\kappa') \,\middle\|\, v\rangle\right\rangle\right\rangle$$

By saturation of $\perp\!\!\!\perp$, it is sufficient to show $\left\langle t \,\middle\|\, \mu\kappa.\left\langle u \,\middle\|\, \mu\kappa'.\langle (\kappa,\kappa') \,\middle\|\, v\rangle\right\rangle\right\rangle \in \perp\!\!\!\perp$. But for each $t' \in \mathbf{H}_{\mathbb{V}}$ and $u' \in \mathbf{G}_{\mathbb{V}}$, by definition of v, one has $\langle (t',u') \,\big\|\, v\rangle \in \perp\!\!\!\perp$. Therefore $\mu\kappa'.\langle (t',\kappa') \,\big\|\, v\rangle \in {\mathbf{G}_{\mathbb{V}}}^\perp$ by lemma 35. Hence $\left\langle u \,\middle\|\, \mu\kappa'.\langle (t',\kappa') \,\middle\|\, v\rangle\right\rangle \in \perp\!\!\!\perp$ by hypothesis on u. Hence $\mu\kappa.\left\langle u \,\middle\|\, \mu\kappa'.\langle (\kappa,\kappa') \,\middle\|\, v\rangle\right\rangle \in {\mathbf{H}_{\mathbb{V}}}^\perp$ by lemma 35. Hence the result by hypothesis on t. (2) and (3): same reasoning. □

Proof (Generation lemma). By induction on the size of A. Case A negative: the result is trivial (by convention). Case $A = R$: $|A| = R^{\perp\perp}$ and is generated by its values since R is a set of values (lemma 36). Case $A = B \otimes C$: $|B| \times |C|$ is equal to ${|B|_{\mathbb{V}}}^{\perp\perp} \times {|C|_{\mathbb{V}}}^{\perp\perp}$ by induction hypothesis, and is

therefore included in $(|B| \times |C|)_{\mathbb{V}}{}^{\perp\perp}$ by lemma 38. Hence $|A|$ is generated by $(|B| \times |C|)_{\mathbb{V}}$. Case $A = B \oplus C$: same proof with $\times$ replaced by $+$. Case $A = !B$: $|A|$ is equal to $(!|B|)^{\perp\perp}$ with $!|B|$ a set of values; hence $|A|$ is generated by its values. Case $A = \mathbf{1}$: A is generated with $\{()\}$; it is therefore generated by its values. Case $A = \mathbf{0}$: A is generated by $\emptyset$; it is therefore generated by its values. Case $A = \exists X\, B$: Similarly to the case $\otimes$, one deduces $(\downarrow |B[R/X]|)_{\mathbb{V}}{}^{\perp} \subseteq (\downarrow |B[R/X]|)^{\perp}$. Therefore:

$$\bigcap_{R\in\Pi} (\downarrow |B[R/X]|)_{\mathbb{V}}{}^{\perp} \subseteq \bigcap_{R\in\Pi} (\downarrow |B[R/X]|)^{\perp}$$

and by basic property of the orthogonal, one has $|\exists X\, B|$ generated by $\bigcup_{R\in\Pi} (\downarrow |B[R/X]|)_{\mathbb{V}}$, which is a set of values. □

Corollary 39. *Let A be a closed formula and $c \in \mathcal{C}$ with $\mathcal{FV}(c) = \{\kappa\}$ where κ is a variable of the same polarity as A. If for all $V \in |A|_{\mathbb{V}}$, one has $c\,[V/\kappa] \in \perp\!\!\!\perp$, then $\mu\kappa.c \Vdash A^{\perp}$.*

Proof. Direct application of lemma 35 and the generation theorem. □

Corollary 40 (Substitution)**.** *Let A a formula with $\mathcal{FV}(A)$ of the form $\{X\}$ and P a closed positive formula. $|P|_{\mathbb{V}}$ is a parameter and one has:*

$$|A\,[P/X]| = \left|A\left[|P|_{\mathbb{V}}/X\right]\right|$$

Proof. By induction on the size of A. Key cases are for A an atom. If $A = X$, then $\left|A\left[|P|_{\mathbb{V}}/X\right]\right| = \left||P|_{\mathbb{V}}\right| = |P|_{\mathbb{V}}{}^{\perp\perp}$, and similarly for $A = X^{\perp}$. One concludes with the theorem of generation. □

H.3. Adequacy lemma

Theorem 41 (Adequacy Lemma, $\mathsf{LK}_{\mathrm{pol}}$)**.** *Let c a command (respectively t a term) typable in LK_{pol}, of type $\vdash \Gamma$ (resp. of type $\vdash t : B \mid \Gamma$) where $\Gamma = \kappa_1 : A_1, \ldots, \kappa_n : A_n$. These formulae have $X_1, \ldots, X_m$ as their free variables. Let $R_1, \ldots, R_m \in \Pi$ be parameters and $u_1, \ldots, u_n$ be closed terms. One writes $\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i\right]$ the substitution $[u_1/\kappa_1, \ldots, u_n/\kappa_n]$ and $\left[\overrightarrow{R}_j \middle/ \overrightarrow{X}_j\right]$ the substitution $[R_1/X_1, \ldots, R_m/X_m]$; one writes for all $i \leq n$, $A'_i = A_i\left[\overrightarrow{R}_j \middle/ \overrightarrow{X}_j\right]$, which is closed. If $u_1 \Vdash A'_1{}^{\perp}, \ldots, u_n \Vdash A'_n{}^{\perp}$, then $c\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i\right] \in \perp\!\!\!\perp$ (resp. $t\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i\right] \Vdash B'$ where $B' = B\left[\overrightarrow{R}_j \middle/ \overrightarrow{X}_j\right]$).*

Proof. By induction on the derivation of c and t. (Activation) $\vdash \mu\kappa.c : B \mid \Gamma$ comes from $c : (\vdash \kappa : B \mid \Gamma)$. This follows from the induction hypothesis and corollary 39. (Cut) $\langle t \,\|\, t' \rangle : (\vdash \Gamma, \Gamma')$ comes from $\vdash t : B \mid \Gamma$ and from $\vdash t' : B^{\perp} \mid \Gamma'$, with $\Gamma = \left\{\kappa_j : A_j \,\middle|\, j \leq i\right\}$ and $\Gamma' = \left\{\kappa_j : A_j \,\middle|\, j > i\right\}$ for a given $0 \leq i \leq n$. One has by induction hypothesis $t\left[\overrightarrow{u}_{j\leq i} \middle/ \overrightarrow{\kappa}_{j\leq i}\right] \Vdash B'$ and $t'\left[\overrightarrow{u}_{j>i} \middle/ \overrightarrow{\kappa}_{j>i}\right] \Vdash B'^{\perp}$. Since $\left|B^{\perp}\left[\overrightarrow{R}_j \middle/ \overrightarrow{X}_j\right]\right| = \left|B\left[\overrightarrow{R}_j \middle/ \overrightarrow{X}_j\right]\right|^{\perp}$, one has $\langle t \,\|\, t' \rangle \left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i\right] \in \perp\!\!\!\perp$. (Weakening, contraction) Trivial. (Tensor) $\vdash (t, u) : B \otimes C \mid \Gamma, \Gamma'$ comes from $\vdash t : B \mid \Gamma$ and from $\vdash u : C \mid \Gamma'$. The result comes from $|B'| \times |C'| \subseteq |B' \otimes C'|$. (Plus) Case similar to the tensor. (Par) $\vdash \mu(\kappa, \kappa').c : B \parr C \mid \Gamma$ comes from $c : (\vdash \kappa : B, \kappa' : C, \Gamma)$. Let $(V, V') \in |B'^{\perp}|_{\mathbb{V}} \times |C'^{\perp}|_{\mathbb{V}}$. One takes $c' = c\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i\right]$. One has $c'\left[V, V' / \kappa, \kappa'\right] \in \perp\!\!\!\perp$ by induction hypothesis. Hence $\langle (V, V') \,\|\, \mu(\kappa, \kappa').c' \rangle \in \perp\!\!\!\perp$ by saturation. Hence $\mu(\kappa, \kappa').c' \in (|B'^{\perp}| \times |C'^{\perp}|)_{\mathbb{V}}{}^{\perp}$. The latter is included in $(|B'^{\perp}| \times |C'^{\perp}|)^{\perp}$ by the generation theorem and by lemma 38. (With) Case similar to the par. (Extraction) $\vdash \{t\} : \exists X\, B \mid \Gamma$ comes from $\vdash t : B[P/X] \mid \Gamma$, with X distinct from $X_1, \ldots, X_m$. One takes $P' = P\left[\overrightarrow{R}_j \middle/ \overrightarrow{X}_j\right]$. By the corollary of substitution 40, one has $\left|B'\left[P'/X\right]\right| = \left|B'\left[|P'|_{\mathbb{V}}/X\right]\right|$. By induction hypothesis, one has $t\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i\right] \Vdash B'\left[P'/X\right]$. Hence $\{t\}\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i\right] \in \downarrow \left|B'\left[|P'|_{\mathbb{V}}/X\right]\right|$. Hence $\{t\}\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i\right] \Vdash \exists X\, B'$. (Generalisation) $\vdash \mu\{\kappa\}.c : \forall X\, B \mid \Gamma$ comes from $c : (\vdash \kappa : B \mid \Gamma)$, with X distinct from $X_1, \ldots, X_m$. By a basic property of the orthogonal, one has $|\forall X\, B'| = \bigcap_{R\in\Pi} \left(\downarrow \left|B'\left[R/X\right]\right|\right)^{\perp}$. Let $R \in \Pi$ and $\{V\} \in \downarrow \left|B'\left[R/X\right]\right|_{\mathbb{V}}$. One has $\langle \{V\} \,\|\, \mu\{\kappa\}.c' \rangle \to c'\left[V/\kappa\right] \in \perp\!\!\!\perp$ by induction hypothesis, hence $\mu\{\kappa\}.c' \in \left(\downarrow \left|B'\left[R/X\right]\right|\right)_{\mathbb{V}}{}^{\perp}$. This is included in $\left(\downarrow \left|B'\left[R/X\right]\right|\right)^{\perp}$ by the generation theorem and lemma 38. ($\mathbf{1}$, $\bot$ and $\top$) Trivial. □

Remark 42. The adequacy lemma holds if one substitutes "LL" for "$\mathsf{LK}_{\mathrm{pol}}$". The proof shall then be extended with the following:

Proof (Adequacy lemma, LL). (Dereliction) $\vdash \text{`}(t) : ?B \mid \kappa_1 : A_1, \ldots, \kappa_n : A_n$ comes from $\vdash t : B \mid \kappa_1 : A_1, \ldots, \kappa_n : A_n$. One takes $t' = t\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i\right]$. Let $\mu\text{`}(\kappa).c \in !\left|B'^{\perp}\right|$. One has by definition $V \in |B'| \Rightarrow c\,[V/\kappa] \in \perp\!\!\!\perp$. If t' is a value, then $\langle \mu\text{`}(\kappa).c \,\|\, \text{`}(t') \rangle \to c\,[t'/\kappa] \in \perp\!\!\!\perp$ since by induction hypothesis $t' \in |B'|$, hence the result. Otherwise, one has $\langle \mu\text{`}(\kappa).c \,\|\, \text{`}(t') \rangle \to \left\langle t' \,\middle\|\, \mu\kappa'.\langle \mu\text{`}(\kappa).c \,\|\, \text{`}(\kappa') \rangle \right\rangle$ with $\mu\kappa'.\langle \mu\text{`}(\kappa).c \,\|\, \text{`}(\kappa') \rangle \in \left|B'^{\perp}\right|$, hence the result with $t' \in |B'|$. (Promotion) $\vdash \mu\text{`}(\kappa).c : !B \mid \kappa_1 : ?A_1, \ldots, \kappa_n : ?A_n$ comes from $c : (\vdash \kappa : B, \kappa_1 : ?A_1, \ldots, \kappa_n : ?A_n)$. One has by induction hypothesis, for each $t \in |B'^{\perp}|$, $c\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i\right][t/\kappa] \in \perp\!\!\!\perp$. Hence $\mu\text{`}(\kappa).c\left[\overrightarrow{u}_i \middle/ \overrightarrow{\kappa}_i\right] \in !|B'|$. □

References

[AH03] Zena M. Ariola and Hugo Herbelin. Minimal classical logic and control operators. In *ICALP '03*, volume 2719 of *LNCS*, pages 871–885. Springer, 2003.

[And92] Jean-Marc Andreoli. Logic programming with focusing proof in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

[BD03] Emmanuel Beffara and Vincent Danos. Disjunctive normal forms and local exceptions. In *ACM SIGPLAN Int. Conf. Func. Prog.*, pages 203–211, Uppsala, Sweden, 2003.

[CH00] Pierre-Louis Curien and Hugo Herbelin. The duality of computation. *ACM SIGPLAN Notices*, 35(9):233–243, 2000.

[CMM10] Pierre-Louis Curien and Guillaume Munch-Maccagnoni. The duality of computation under focus. In *IFIP TCS*, 2010.

[DJS95] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. A new deconstructive logic: Linear logic. *Journal of Symbolic Logic*, 62 (3):755–807, 1995.

[DL06] Roy Dyckhoff and Stéphane Lengrand. LJQ, a strongly focused calculus for intuitionistic logic. In *Proceedings of the 2nd Conference on Computability in Europe (CiE'06)*, 2006.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir91] Jean-Yves Girard. A new constructive logic: Classical logic. *Math. Struct. Comp. Sci.*, (1), 1991.

[Gir93] Jean-Yves Girard. On the unity of logic. *Ann. Pure Appl. Logic*, 59(3):201–217, 1993.

[Gir01] Jean-Yves Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11:301–506, 2001.

[Gir07] Jean-Yves Girard. *Le Point Aveugle, Cours de logique, Tome II: Vers l'imperfection*. Visions des Sciences. Hermann, 2007.

[Her05] Hugo Herbelin. C'est maintenant qu'on calcule, au cœur de la dualité, 2005. Habilitation thesis.

[Her08] Hugo Herbelin. Duality of computation and sequent calculus: a few more remarks. Manuscript, 2008.

[Kri93] Jean-Louis Krivine. *Lambda-calculus, types and models*. Ellis Horwood, 1993.

[Kri04] Jean-Louis Krivine. Realizability in classical logic. To appear in *Panoramas et synthèses*, Société Mathématique de France, 2004.

[Kri08] Jean-Louis Krivine. Structures de réalisabilité, RAM et ultrafiltre sur N. To appear, 2008.

[Lau02] Olivier Laurent. *Etude de la polarisation en logique*. Thèse de doctorat, Université Aix-Marseille II, mar 2002.

[LM08] Stéphane Lengrand and Alexandre Miquel. Classical F_ω, orthogonality and symmetric candidates. *Ann. Pure Appl. Logic*, 153(1-3):3–20, 2008.

[MM08] Guillaume Munch-Maccagnoni. Étude polarisée du système L (v3). Master's thesis, July 2008.

[Nip91] Tobias Nipkow. Higher-order critical pairs. In *Proc. 6th IEEE Symp. Logic in Computer Science*, pages 342–349. IEEE Press, 1991.

[Ter08] Kazushige Terui. Computational Ludics, 2008. To appear in TCS.

[Wad03] Philip Wadler. Call-by-value is dual to call-by-name. *SIGPLAN Not.*, 38(9):189–201, 2003.

[Zei08] Noam Zeilberger. On the unity of duality. *Ann. Pure and App. Logic*, 153:1, 2008.

[Zei09] Noam Zeilberger. Refinement types and computational duality. PLPV '09, 2009.